TUGAS AKHIR – KI141502

CODEREPLAYER: PEMUTAR ULANG PROSES PEMBUATAN PROGRAM UNTUK MENDUKUNG PEMBELAJARAN PEMROGRAMAN

Muhammad Farhan Adha
NRP 5111 100 109

Dosen Pembimbing
Umi Laili Yuhana, S.Kom., M.Sc.
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KI141502

CODEREPLAYER: CODING PROCESS REPLAYER TO SUPPORT PROGRAMMING LEARNING

Muhammad Farhan Adha
NRP 5111 100 109

Advisor
Umi Laili Yuhana, S.Kom., M.Sc.
Rizky Januar Akbar, S.Kom., M.Eng.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

LEMBAR PENGESAHAN

CodeReplayer: Pemutar Ulang Proses Pembuatan Program untuk Mendukung Pembelajaran Pemrograman

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
MUHAMMAD FARHAN ADHA
NRP : 5111 100 109

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Umi Laili Yuhana, S.Kom., M.Sc. .....
NIP: 19790626 200501 2 002 (pembimbing 1)

Rizky Januar Akbar, S.Kom., M.Eng. .....
NIP: 19870103 201404 1 001 (pembimbing 2)

SURABAYA
JUNI 2015

CodeReplayer: Pemutar Ulang Proses Pembuatan Program untuk Mendukung Pembelajaran Pemrograman

Nama Mahasiswa	: Muhammad Farhan Adha
NRP	: 5111 100 109
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Umi Laili Yuhana, S.Kom., M.Sc.
Dosen Pembimbing 2	: Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRAKSI

Proses belajar pemrograman adalah tahap yang akan dilalui ketika seseorang mempelajari hal baru di bidang pemrograman. Dibutuhkan metode belajar yang tepat agar proses belajar tidak terlalu banyak menguras waktu, biaya, dan usaha. Metode umum yang biasa digunakan ketika belajar pemrograman adalah dengan membaca kode sumber atau implementasi dari program yang dipelajari. Namun, hal tersebut memiliki kelemahan karena kode sumber bersifat statis sedangkan pembuatan kode sumber adalah serangkaian proses dinamis yang cenderung ditulis tidak berurutan dari baris satu ke baris selanjutnya.

Dengan adanya permasalahan tersebut, dalam tugas akhir ini dibuat sebuah aplikasi berbasis *website* untuk dapat melakukan reka ulang pembuatan kode sumber secara kronologis, CodeReplayer. Aplikasi ini memanfaatkan *plug-in* OperationRecorder, yang terintegrasi dengan IDE (*Integrated Development Environment*) Eclipse dan berfungsi untuk melakukan perekaman operasi pada pembuatan kode sumber. Selanjutnya, CodeReplayer akan mengolah hasil perekaman tersebut hingga dapat dilihat hasil reka ulang pembuatan kode sumber sesuai dengan urutan penulisannya.

CodeReplayer diuji dengan dua metode pengujian yaitu *white box* dan *black box.* Pada pengujian *white box,* digunakan kakas

bantu berupa CodeMetrics dan NDepend yang menghasilkan *output* berupa ukuran-ukuran untuk menguji kualitas kode program CodeReplayer. Sedangkan pada pengujian *black box*, dilakukan pengujian pada tiap fungsional sistem dengan beberapa skenario kasus uji. Dari hasil pengujian *black box* tersebut, didapatkan hasil bahwa semua kebutuhan fungsional sistem berjalan dengan baik dan menghasilkan *output* sesuai dengan yang diharapkan.

Kata kunci: CodeReplayer, OperationRecorder, Kode Sumber.

CodeReplayer: Coding Process Replayer to Support Programming Learning

Student Name : Muhammad Farhan Adha
NRP : 5111 100 109
Major : Teknik Informatika FTIf-ITS
Advisor 1 : Umi Laili Yuhana, S.Kom., M.Sc.
Advisor 2 : Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRACT

Learning programming is a process when someone wants to start or learn something new in programming. To save time, cost, and effort a new method is needed to learn programming. Normally, people read from the source code when they are learning programming. This method has a weakness because source code does not track every change when the code is written. Source code only captures the final result from coding process.

To address these problems, this final project develops a web-based software which has feature to replay coding process chronologically, named CodeReplayer. This software uses OperationRecorder plug-in which is integrated with IDE (Integrated Development Environment) Eclipse and has a function to record the operation chronologically and automatically when someone writes the code. Then, CodeReplayer processes OperationRecorder recording results to replay coding process chronologically.

CodeReplayer is tested by white box and black box testing. The white box uses CodeMetrics and NDepend to determine CodeReplayer's source code quality and The black box tests each functional requirement with some testing scenarios. The result of black box testing indicated that CodeReplayer's functionals are running well as expected.

***Keywords**: CodeReplayer, OperationRecorder, Source Code.*

KATA PENGANTAR

Puji syukur Alhamdulillah kepada Allah Yang Maha Kuasa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

"*CodeReplayer: Pemutar Ulang Proses Pembuatan Program untuk Mendukung Pembelajaran Pemrograman*"

Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Ibu Badriah, Bapak (Alm) Abusamah, Bapak Sugiarto, Pakde Taufik, kakak Noprilia, adik Cahyo Kumara, dan keluarga besar yang selalu memberikan dukungan penuh selama masa studi dan penyelesaian tugas akhir ini.
2. Ibu Umi Laili Yuhana dan Bapak Rizky Januar Akbar sebagai dosen pembimbing yang telah bersedia meluangkan waktu, tenaga, serta pikirannya untuk mendukung penulis selama proses pengerjaan tugas akhir ini.
3. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak berbagi ilmu dan pengalamannya yang tak ternilai harganya bagi penulis selama masa studi di Teknik Informatika ITS.
4. Teman-teman TC 2011, Askary, Ilmi, Alrezza, Hayam, Syahdeini dan anggota Morning On Fire (MorOn) lainnya yang selalu bisa menjadi penghibur dan pelepas penat ditengah sibuknya perkuliahan.
5. Keluarga Besar Admin LP yang sudah menjadi tempat bernaung selama masa studi penulis di Teknik Informatika ITS.
6. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu -persatu.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian pada tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2015

Muhammad Farhan Adha

DAFTAR ISI

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR GAMBAR

LAMPIRAN A. KODE SUMBER

```
<script>
    function showHideProp() {
        if
(document.getElementById('prop').style.visibility ==
'collapse') {
            $('#prop').css('visibility', 'visible');

document.getElementById('buttonShow').style.backgroun
dImage = "url(/Picture/arrowup.png)";

        }
        else {
            $('#prop').css('visibility', 'collapse');

document.getElementById('buttonShow').style.backgroun
dImage = "url(/Picture/arrowdown.png)";
        }
    }

    $("#number").selectmenu({
        change: function () {
            var selectedIndex = $("#number").val();

document.getElementById("divauto").style.fontSize =
selectedIndex + "px";
        }
    });

    $('#fontSelect').fontSelector({
        'hide_fallbacks': true,
        'initial': 'Courier New,Courier
New,Courier,monospace',
        'selected': function (style) {

document.getElementById("divauto").style.fontFamily =
style;
        },
        'fonts': [
```

```
        'Arial,Arial,Helvetica,sans-serif',
        'Arial Black,Arial Black,Gadget,sans-serif',
        'Comic Sans MS,Comic Sans MS,cursive',
        'Courier New,Courier New,Courier,monospace',
        'Georgia,Georgia,serif',
        'Impact,Charcoal,sans-serif',
        'Lucida Console,Monaco,monospace',
        'Lucida Sans Unicode,Lucida Grande,sans-
serif',
        'Palatino Linotype,Book
Antiqua,Palatino,serif',
        'Tahoma,Geneva,sans-serif',
        'Times New Roman,Times,serif',
        'Trebuchet MS,Helvetica,sans-serif',
        'Verdana,Geneva,sans-serif',
        'Gill Sans,Geneva,sans-serif'
        ]
    });
    $('#font_color_picker').colpick({
        colorScheme: 'dark',
        layout: 'rgbhex',
        color: 'F0F0F0',
        onSubmit: function (hsb, hex, rgb, el) {
            $(el).css('background-color', '#' + hex);

document.getElementById("divauto").style.color = '#'
+ hex;
            $(el).colpickHide();
        }
    })
.css('background-color', '#F0F0F0');

    $('#bg_color_picker').colpick({
        colorScheme: 'dark',
        layout: 'rgbhex',
        color: '000000',
        onSubmit: function (hsb, hex, rgb, el) {
            $(el).css('background-color', '#' + hex);
            document.getElementById("text-
body").style.backgroundColor = '#' + hex;
```

```
            $(el).colpickHide();
        }
    })
.css('background-color', '#000000');

    Element.prototype.documentOffsetTop = function ()
{
        return this.offsetTop + (this.offsetParent ?
this.offsetParent.documentOffsetTop() : 0);
    };

    var stop = 0;
    var pause = 0;
    var pauseCount = 0;
    var to = 0;
    var isFFOn = 0;
    var node;
    var lastNode;
    var charAt2;
    var lastChar;
    var resumeOffset;
    var len;
    var resp;
    var startNode;
    var actionUrl;
    var speedChar;
    var showNote;


    window.onload = doOnLoad;

    function doOnLoad() {
        actionUrl = '@Url.Action("GetJsonOperations",
"SpeedOptions", new { refNum = @Model })';
        $.getJSON(actionUrl, displayData);

document.getElementById('prop').style.visibility =
'collapse';
    }
```

```
    $("#btnPlay").click(function () {
        var pc;
        visibilityIcon(false, true, false, false);

        if (pause == 1 && stop == 0) {
            pause = 0;
            resumeTyping();
        }
        else if (stop > 0) {
            if (stop == 1)
                pause = 0;
            else if (stop == 2) {

document.getElementById("divauto").innerHTML = "";
                displayData(resp);
            }
            stop = 0;
            pc = pauseCount;
        }
        if (pauseCount == 0 || pc == pauseCount) {
            isFFOn = 0;
            startPlay();
        }
    })

    $("#btnPause").click(function () {
        visibilityIcon(false, false, true, true);
        pauseCount++;
        pause = 1;
    })

    $("#btnFastForward").click(function () {
        visibilityIcon(false, true, false, false);
        pauseCount++;
        if (isFFOn == 0) {

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-red.png)";
            speedChar /= 2;
            isFFOn = 1;
```

```
        }
        else {

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
            speedChar *= 2;
            isFFOn = 0;
        }
        resumeTyping();
    })

    $("#btnStop").click(function () {
        pauseCount++;
        stop = 1;

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
        visibilityIcon(true, false, true, true);
        document.getElementById("divauto").innerHTML
= "";

document.getElementById("noteDialog").innerHTML = "";
        displayData(resp);
    })

    function visibilityIcon(stop, play, pause,
fastForward) {
        document.getElementById("btnStop").disabled =
stop;
        document.getElementById("btnPlay").disabled =
play;
        document.getElementById("btnPause").disabled
= pause;

document.getElementById("btnFastForward").disabled =
fastForward;
    }

    function stopClear() {
        stop = 2;
```

```
document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
        visibilityIcon(true, false, true, true);
    }

    function createSpan(num) {
        for (var i = 0; i <= num; i++) {
            $('#divauto').append("<span
class='spanChar' id=" + i + "></span>");
        }
    }

    function initStartCode(response) {
        var tempStartNode;
        for (var i = 0; ; i++) {
            if (response.Operations[i].Time != 0) {
                tempStartNode = i;
                break;
            }
            var offs = response.Operations[i].Offset;
            var txt =
response.Operations[i].Character;
            for (var n = 0; n < txt.length; n++) {
                if (txt[n] == '#' && txt[n + 1] ==
'n' && txt[n + 2] == '#') {
                    $('#' + offs).append("<br/>");
                    offs += 2;
                    n += 2;
                }
                else if (txt[n] == '#' && txt[n + 1]
== 't' && txt[n + 2] == '#') {
                    $('#' +
offs).append("   ");
                    offs += 1;
                    n += 2;
                } else if (txt[n] == '#' && txt[n +
1] == 's' && txt[n + 2] == '#') {
                    $('#' + offs).append(" ");
                    offs += 1;
```

```
                    n += 2;
                }
                else {
                    $('#' + offs).append(txt[n]);
                    offs++;
                }
            }
        }
        startNode = tempStartNode;
    }

    function displayData(response) {
        resp = response;
        document.getElementById("titleCR").innerHTML
= "Code Replayer";
        document.getElementById("titleCR").innerHTML
+= " - " + response.Title;
        maxSpan = response.MaxOffset;
        lastNode = resp.Operations.length - 1;
        lastChar =
resp.Operations[lastNode].Character.length - 1;
        createSpan(maxSpan);
        initStartCode(response);
    }

    function startPlay() {
        var pc = pauseCount;
        var nextTo;
        speedChar = resp.Speed;
        to = 300;
        len = resp.Operations.length;

        for (var j = startNode; j < len; j++) {
            appendNoteDialog(j, pc);
            if (resp.Operations[j].OperationName ==
"inserted")
                inserted(j, 0, to,
resp.Operations[j].Character,
resp.Operations[j].Offset, pc);
```

```
            else if (resp.Operations[j].OperationName
== "deleted")
                deleted(j,
resp.Operations[j].Character.length - 1, to,
resp.Operations[j].Character,
resp.Operations[j].Offset - 1, pc);
            if (j < len - 1) {
                nextTo =
(resp.Operations[j].CurrDeltaOffset -
resp.Operations[j].LineNumber) * speedChar;
                to += (resp.Operations[j + 1].Time +
nextTo);
            }
        }
    }


    function resumeTyping() {

        var pc = pauseCount;
        var nextTo;
        to = 0;

        appendNoteDialog(node, pc);
        if (resp.Operations[node].OperationName ==
"inserted")
            inserted(node, charAt2 + 1, to,
resp.Operations[node].Character, resumeOffset, pc);
        else if (resp.Operations[node].OperationName
== "deleted")
            deleted(node, charAt2 - 1, to,
resp.Operations[node].Character, resumeOffset - 1,
pc);
        if (node < len - 1) {
            nextTo =
(resp.Operations[node].CurrDeltaOffset -
resp.Operations[node].LineNumber) * speedChar;
            to += (resp.Operations[node + 1].Time +
nextTo);
        }
```

```
        for (var j = node + 1; j < len; j++) {
            appendNoteDialog(j, pc);
            if (resp.Operations[j].OperationName ==
"inserted")
                inserted(j, 0, to,
resp.Operations[j].Character,
resp.Operations[j].Offset, pc);
            else if (resp.Operations[j].OperationName
== "deleted")
                deleted(j,
resp.Operations[j].Character.length - 1, to,
resp.Operations[j].Character,
resp.Operations[j].Offset - 1, pc);
            if (j < len - 1) {
                nextTo =
(resp.Operations[j].CurrDeltaOffset -
resp.Operations[j].LineNumber) * speedChar;
                to += (resp.Operations[j + 1].Time +
nextTo);
            }
        }
    }

    function appendNoteDialog(num, pc) {
        showNote = setTimeout(function () {
            if (pc == pauseCount)

document.getElementById("noteDialog").innerHTML =
resp.Operations[num].Note;
        }, to);
    }

    function inserted(j, charAt, timeOut, txt,
offset, pc) {
        var delay = speedChar;
        var to = timeOut;
        offset = parseInt(offset);
        for (var i = charAt; i < txt.length; i++) {
```

```
            var temp = doInsert(j, i, txt, to,
offset, pc);
            i = temp[0];
            offset = temp[1];
            offset++;
            if (temp[2] == 1)
                offset++;
            to += delay;
        }
    }

    function doInsert(j3, n, txt, to, offs, pc) {
        var nl = 0;

        if (txt[n] == '#' && txt[n + 1] == 'n' &&
txt[n + 2] == '#') {
            setTimeout(function () {
                insertNewline(j3, n, offs, pc);
            }, to)
            n += 2;
            nl = 1;
        }
        else if (txt[n] == '#' && txt[n + 1] == 't'
&& txt[n + 2] == '#') {
            setTimeout(function () {
                insertTab(j3, n, offs, pc);
            }, to)
            n += 2;
        } else if (txt[n] == '#' && txt[n + 1] == 's'
&& txt[n + 2] == '#') {
            setTimeout(function () {
                insertSpace(j3, n, offs, pc);
            }, to)
            n += 2;
        }
        else {
            setTimeout(function () {
                insertOtherChar(j3, n, offs, txt[n],
pc);
            }, to)
```

```
        }

        return [n, offs, nl];
    }

    function insertNewline(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append("<br/>");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 2;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function insertTab(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' +
offs).append("    ");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }
```

```
    function insertSpace(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append(" ");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function insertOtherChar(j4, n2, offs, c, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append(c);
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function deleted(j, charAt, timeOut, txt, offset,
pc) {
        var delay = speedChar;
        var to = timeOut;
        offset = parseInt(offset);
        for (var i = charAt; i >= 0; i--) {
```

```
            var temp = doDelete(j, i, txt, to,
offset, pc);
            i = temp[0];
            offset = temp[1];
            offset--;
            if (temp[2] == 1)
                offset--;
            to += delay;
        }
    }

    function doDelete(j3, n, txt, to, offs, pc) {
        var nl = 0;
        if (txt[n] == '#' && txt[n - 1] == 'n' &&
txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);
                document.getElementById(String(offs -
1)).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs - 1;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
            nl = 1;
        }
        else if (txt[n] == '#' && txt[n - 1] == 't'
&& txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
```

```
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
        }
        else if (txt[n] == '#' && txt[n - 1] == 's'
&& txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
        }
        else {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
```

```
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
        }
        return [n, offs, nl];
    }

</script>
```

Kode Sumber 8.1 Javascript proses reka ulang

BAB I
PENDAHULUAN

Pada bab ini dipaparkan mengenai garis besar tugas akhir, meliputi latar belakang, tujuan, rumusan permasalahan, batasan permasalahan, metodologi penyelesaian tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Proses belajar pemrograman adalah tahap yang akan dilalui ketika seseorang belajar hal baru di bidang pemrograman. Proses belajar pemrograman dapat memakan waktu dan biaya yang relatif pada beberapa faktor seperti metode, pengalaman, kemampuan *programming*, dan lain sebagainya.

Pengalaman dan kemampuan *programming* akan tumbuh dan berkembang seiring berjalannya waktu dan latihan-latihan pemrograman. Sedangkan metode adalah hal teknis yang diterapkan sesuai dengan keinginan subjek untuk menunjang kemudahan dalam belajar pemrograman. Namun, metode belajar yang kurang tepat dapat memperlambat proses belajar sehingga akan lebih banyak sumber daya seperti waktu, biaya, dan usaha yang terkuras.

Metode umum yang biasa digunakan ketika belajar pemrograman adalah dengan membaca kode sumber atau implementasi dari program yang dipelajari. Namun, hal tersebut memiliki kelemahan karena kode sumber bersifat statis sedangkan penulisan kode sumber adalah serangkaian proses dinamis yang cenderung ditulis tidak berurutan dari baris satu ke baris selanjutnya. Maka dari itu, dibutuhkan solusi yang dapat mempertemukan antara subjek dan kode sumber dengan cara yang lebih baik.

Dengan adanya permasalahan tersebut, dalam tugas akhir ini dibuat sebuah aplikasi berbasis *website* untuk dapat melakukan reka ulang pembuatan kode sumber secara kronologis, bernama CodeReplayer. Aplikasi ini memanfaatkan *plug-in*

OperationRecorder, yang terintegrasi dengan IDE (Integrated Development Environment) Eclipse dan berfungsi untuk melakukan perekaman operasi pada pembuatan kode sumber. Selanjutnya, CodeReplayer akan mengolah hasil perekaman tersebut hingga dapat dilihat hasil reka ulang pembuatan kode sumber sesuai dengan urutan penulisannya.

Dengan pembuatan tugas akhir ini, diharapkan dapat menjadi solusi yang lebih baik di bidang edukasi khususnya untuk belajar dan mengajar pemrograman. Tentunya, jika CodeReplayer mampu secara efektif menjadi metode dalam proses belajar dan mengajar pemrograman, maka akan berdampak pada berkurangnya biaya, waktu, dan usaha yang dibutuhkan untuk mempelajari pemrograman.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini sebagai berikut:

1. Bagaimana mengolah berkas XML hasil dari perekaman pembuatan kode program oleh *plug-in* OperationRecorder agar lebih mudah dimengerti?
2. Bagaimana melakukan reka ulang dari proses pembuatan kode program secara tepat dan akurat?
3. Bagaimana melakukan penyuntingan waktu jeda pada berkas XML hasil dari perekaman *plug-in* OperationRecorder?

1.3. Batasan Permasalahan

Beberapa batasan dalam tugas akhir ini adalah sebagai berikut:

1. CodeReplayer merupakan aplikasi berbasis *web*.
2. IDE yang mendukung untuk perekaman kode adalah Eclipse dengan bantuan *plug-in* OperationRecorder di dalamnya.
3. Hasil dari proses merekam dengan Eclipse dapat diputar kembali pada halaman *web* CodeReplayer.

4. Dalam proses pemutaran ulang, fitur yang tersedia adalah: *play*, *pause*, *stop*, dan *fast forward* dua kali.
5. *Plug-in* OperationRecorder akan mencatat aktivitas terkait penyuntingan seperti menambah, menghapus, memodifikasi, dan melakukan penyalinan karakter.
6. Aktivitas yang tidak direkam oleh OperationRecorder seperti perpindahan kursor, seleksi teks, membuka dan menutup berkas, berpindah berkas, dan mengganti nama berkas.

1.4. Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Membuat aplikasi yang mampu mengolah berkas XML hasil dari perekaman pembuatan kode program oleh *plug-in* OperationRecorder agar lebih mudah dimengerti.
2. Membuat aplikasi yang mampu melakukan reka ulang dari proses pembuatan kode program secara tepat dan akurat.
3. Membuat aplikasi yang dapat melakukan penyuntingan waktu jeda pada berkas XML hasil dari perekaman *plug-in* OperationRecorder.

1.5. Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Menjadi solusi yang lebih baik sebagai alat bantu dalam memahami kode program.
2. Membuat hasil perekaman *plug-in* OperationRecorder lebih mudah dipahami.

1.6. Metodologi

a) Penyusunan proposal tugas akhir

Proposal tugas akhir ini akan mendeskripsikan dan membahas mengenai rencana pengembangan aplikasi CodeReplayer yang memanfaatkan hasil perekaman dari *plug-in* OperationRecorder pada IDE Eclipse dan melakukan reka ulang dari hasil perekaman

tersebut. Secara detil, proposal tugas akhir ini dibagi ke dalam beberapa bagian, yaitu: latar belakang diajukannya tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir.

Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir dan ringkasan isi yang membahas metode yang akan digunakan dalam tugas akhir Subbab metodologi merupakan penjelasan mengenai tahapan penyusunan buku tugas akhir. Terdapat pula subbab jadwal pengerjaan yang menjelaskan jadwal pengerjaan tugas akhir dan di akhir bagian terdapat daftar pustaka untuk mencantumkan referensi yang digunakan dalam tugas akhir.

b) Studi literatur

Pada tugas akhir ini, akan dipelajari sejumlah referensi yang diperlukan sebagai ilmu penunjang dalam pembuatan aplikasi. Adapun literatur yang dipakai adalah:

1. Teori konsep dan penggunaan *plug-in* OperationRecorder pada IDE Eclipse
2. Teori konsep dan penggunaan *plug-in* OperationReplayer pada IDE Eclipse
3. Teori konsep *MVC(Model, View, Control) Pattern*
4. Teori konsep ASP.NET
5. Teori konsep Bootstrap
6. Teori konsep ASP.NET MVC 5
7. Teori konsep XML *Document*
8. Teori penggunaan XMLSerializer pada ASP.NET MVC 5
9. Teori penggunaan Javascript *Timing Events*
10. Teori penggunaan Jquery dan JSON pada ASP.NET MCV5

c) Analisis dan desain perangkat lunak

Pada aplikasi CodeReplayer, yang bertindak sebagai aktor adalah pengguna IDE Eclipse yang memanfaatkan *plug-in*

OperationRecorder. Adapun beberapa fitur dari CodeReplayer adalah sebagai berikut:

1. Melakukan unggah berkas XML yang merupakan hasil perekaman *plug-in* OperationRecorder pada IDE Eclipse.
2. Mengolah berkas XML yang diunggah agar bisa dilakukan pemutaran ulang proses pembuatan kode program.
3. Memainkan ulang hasil rekam proses pembuatan kode program pada halaman *web* CodeReplayer.

d) Implementasi perangkat lunak

Berikut beberapa alat bantu yang diperlukan dalam implementasi aplikasi:

1. IDE Eclipse
2. *Plug-in* OperationRecorder pada IDE Eclipse
3. Microsoft Visual Studio 2013
4. Sublime Text 2
5. Java Development Kit
6. Java Runtime Environment
7. *Web browser*

e) Pengujian dan evaluasi

Proses pengujian dari aplikasi ini akan dilakukan secara *black box* dan *white box*. Pengujian *black box* adalah pengujian yang berfokus pada spesifikasi fungsional perangkat lunak. Penguji dapat mendefinisikan kumpulan kasus uji coba dan melakukan pengujian dari kasus uji coba tersebut dengan spesifikasi fungsional program. Sedangkan pengujian *white box* adalah metode pengujian yang berfokus pada kode program untuk mengidentifikasi cacat yang mungkin terjadi pada program.

f) Penyusunan buku tugas akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

g) Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk memberikan gambaran mengenai tugas akhir. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan, rumusan permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa pustaka penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan kebutuhan perangkat lunak.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak dan implementasi dari tiap lapisan *model-view-controller*.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian *black box* dan *white box*.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

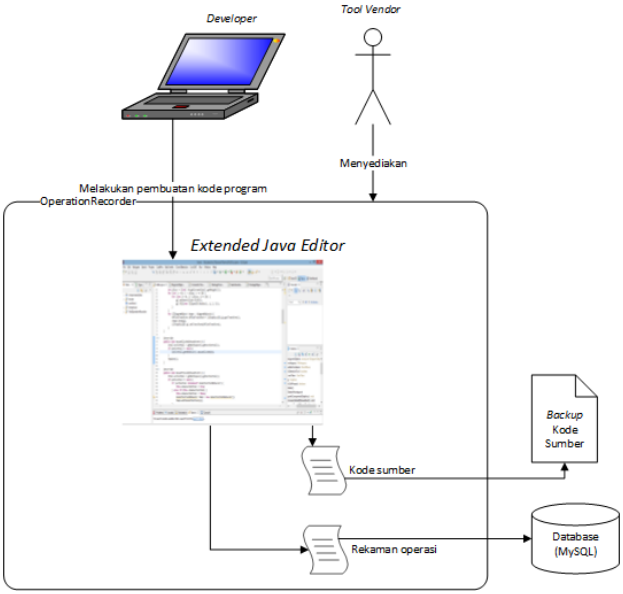
Merupakan bab tambahan yang berisi data tambahan yang penting pada aplikasi ini.

BAB II
TINJAUAN PUSTAKA

Pada bab ini membahas mengenai pustaka yang digunakan dalam pembuatan tugas akhir. Adapun pustaka tersebut meliputi OperationRecorder *plug-in*, MVC (*Model*, *View*, *Controller*) *Pattern*, ASP.NET, Bootstrap, ASP.NET MVC 4, Entity *Framework* 6, JQuery, Json dan lainnya.

2.1. *Plug-in* OperationRecorder

OperationRecorder adalah sebuah *plug-in* yang melakukan pencatatan atau perekaman operasi ketika seseorang melakukan pembuatan kode program pada IDE Eclipse. OperationRecorder melakukan perekaman tersebut dengan memanfaatkan *undo history* pada Eclipse. Arsitektur dari *plug-in* OperationRecorder ditunjukkan pada Gambar 2.1.



Gambar 2.1 Arsitektur OperationRecorder

Untuk mendapatkan hasil perekaman, pengguna tidak harus melakukan *compile* terlebih dahulu. Pengguna cukup menutup jendela editor berkas yang bersangkutan pada Eclipse, maka berkas hasil perekaman operasi akan berada pada folder *history* didalam direktori *project* tersebut.

Operasi yang direkam oleh OperationRecorder meliputi: penulisan, penghapusan, modifikasi karakter, *cut*, dan *copy-paste*. OperationRecorder tidak melakukan perekaman terhadap operasi, seperti: perpindahan kursor, seleksi teks, buka dan tutup berkas, berpindah berkas, mengubah nama berkas, dan *undo-redo*. Detil dari struktur data OperationRecorder ditunjukkan pada Tabel 2.1 [1].

Tabel 2.1 Struktur Data OperationRecorder

Column Name	Description
time	When the operation was performed
xth	Suffix of identifier (ID)
developer	Who performed the operation
file	Changed file in the Eclipse workspace
sort	“inserted”, “deleted”, or “modified”
start	Offset for the starting point of the text
end	Offset for the ending point of the text
inserted	Inserted text
deleted	Deleted text
parent	Reference to a composite record
classNameForward	Class name for the inserted text
methodNameForward	Method name for the inserted text
nodePathForward	AST node for the insertion
classNameBackward	Class name for the deleted text
methodNameBackward	Method name for the deleted text
nodePathBackward	AST node for the deletion

2.2. OperationReplayer *plug-in*

Seperti halnya OperationRecorder, OperationReplayer adalah *plug-in* pada IDE Eclipse. Namun, jika OperationRecorder berfungsi untuk merekam proses pembuatan program, OperationReplayer berfungsi untuk melakukan reka ulang pembuatan program yang memanfaatkan hasil perekaman OperationRecorder [2].

Pada tugas akhir ini, dibuat aplikasi yang juga berfungsi untuk melakukan reka ulang pembuatan program yang memanfaatkan hasil perekaman OperationRecorder yaitu CodeReplayer. Namun terdapat beberapa perbedaan dan keunggulan dari CodeReplayer, yaitu:

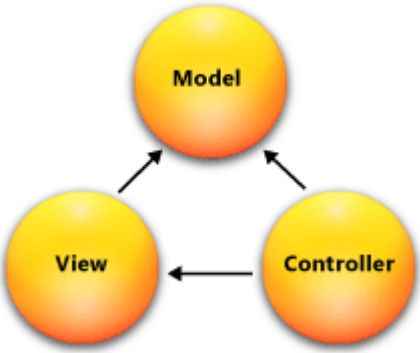
1. CodeReplayer adalah aplikasi berbasis *website* sehingga proses reka ulang pembuatan kode program mendukung berbagai macam *platform* serta tidak memerlukan IDE Eclipse untuk melihat proses reka ulang.
2. Pada CodeReplayer terdapat fitur penyuntingan berkas XML hasil OperationRecorder dengan fitur seperti: menambahkan dokumentasi pada tiap operasi, memilih kecepatan tiap karakter, dan mengurangi atau menghapus jeda waktu antar operasi.
3. Lebih mendukung pada kebutuhan edukasi (*e-learning*) seperti tutorial dengan bantuan fitur *embed* yang memungkinkan pengguna dapat memasang hasil reka ulang pembuatan program di halaman-halaman *website* mereka.

2.3. MVC (Model, View, Controller) Pattern

MVC merupakan arsitektur pembangunan sistem yang memisahkan sistem ke dalam tiga komponen utama, yaitu *model*, *view*, dan *controller*. Arsitektur Model-View-Controller ditunjukkan pada Gambar 2.2.

Pada aturan MVC, data aplikasi ditangani oleh *model* dan presentasi data ditangani oleh *view*. *Model* dan *view* bersifat pasif. Masing-masing menunggu permintaan. Lapisan yang berfungsi untuk memanggil keduanya adalah *controller*. *Controller* adalah

penggerak dan penghubung pada arsitektur MVC. *Controller* menunggu permintaan pengguna. Ketika permintaan tiba, *controller* mengambil beberapa data dari *model* atau memberikan informasi kepada *model* untuk memodifikasi beberapa data. Kemudian *controller* ke *view*. *View* menampilkan data kepada pengguna sesuai dengan permintaan yang diberikan kepada sistem.



Gambar 2.2 Arsitektur Model View Controller

Model berfungsi untuk menyimpan model dari sistem. Untuk data permanen, biasanya disimpan dalam basis data. Komponen ini tidak bisa mengakses kelas *view* dan *controller*. Dia bersifat seperti bagian persistensi dari sistem.

View menyimpan kelas-kelas yang berhubungan dengan antarmuka. Kelas ini yang akan mengakses kelas-kelas yang ada di *model* dan *controller*. Biasanya kelas-kelas ini dibangun dengan HTML. Kelas-kelas ini digunakan untuk menampilkan data yang diambil dari model.

Controller merupakan komponen yang menyimpan kelas-kelas yang mengatur hubungan antara *model* dan *view*. Kelas ini menerima kejadian dari luar, berinteraksi dengan *model* dan menampilkan ke *view* kepada pengguna [3].

2.4. ASP.NET

ASP.NET merupakan *framework* untuk membangun *web.* Pada ASP.NET terdapat tiga teknologi untuk pembangunan aplikasi *web* yaitu ASP.NET Web Form, ASP.NET Web Pages, dan ASP.NET MVC. ASP.NET menjadi sebuah *web platform* yang menyediakan seluruh servis untuk membangun aplikasi *web* yang berbasis pada kelas *enterprise-server*. ASP.NET dibangun pada .NET *Framework* sehingga semua fitur pada .NET *Framework* tersedia pada ASP.NET [4].

2.5. ASP.NET MVC 5

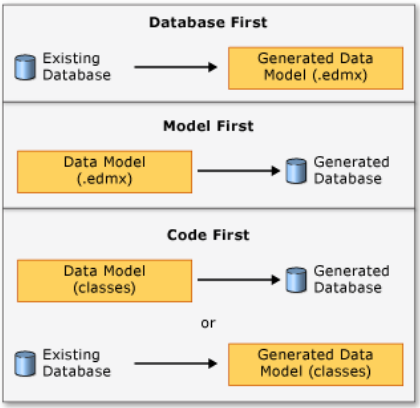
ASP.NET MVC merupakan alternatif dari ASP.NET Web Forms dalam membuat aplikasi berbasis *web*. ASP.NET MVC secara arsitektur menggunakan konsep MVC yang terbagi menjadi 3 komponen utama yaitu *model*, *view*, dan *controller*. MVC *Framework* pada ASP.NET MVC didefinisikan pada *file assembly* **System.Web.Mvc**. ASP.NET MVC juga terintegrasi dengan beberapa fitur ASP.NET yang sudah ada sebelumnya seperti *master pages* dan otentikasi pengguna [3].

2.6. Bootstrap

Bootstrap merupakan *framework* berbasis *open-source* yang merupakan kakas bantu dan digunakan untuk membuat aplikasi *web*. Bootstrap berisi kumpulan *template* HTML dan CSS untuk elemen-elemen HTML seperti *form*, *button*, dan berbagai macam komponen antarmuka lainnya. Tujuan dari Bootstrap *Framework* adalah untuk mempermudah proses pengembangan *web*. Karena itu, Bootstrap *Framework* saat ini sudah sangat populer digunakan hingga pada Maret 2015 GitHub mencatat lebih dari *78,000 stars* dan *30,000 forks* dilakukan pada *project* Bootstrap. Jumlah tersebut tercatat sebagai jumlah *stars* terbanyak sebuah *project* pada situs GitHub [5].

2.7. ADO.NET Entity *Framework* 6

ADO.NET Entity *Framework* adalah sebuah *framework open-source* berbasis *object-relational mapping* (ORM) didalam .NET *Framework*. Entity *Framework* digunakan untuk menyimpan data yang berupa instansiasi dari kelas model kedalam bentuk tabel-tabel relasional. Ada 3 cara pendekatan dalam menggunakan Entity *Framework* sebagai basis data, yaitu: *Database First, Model First, dan Code First* [6]. Perbadaan ketiganya ditunjukkan pada Gambar 2.3. Pada tugas akhir ini, digunakan pendekatan dengan tipe *Code First*.



Gambar 2.3 Berbagai metode implementasi Entity Framework

2.8. XmlDocument dan XmlSerializer

XmlDocument dan XmlSerializer merupakan kelas yang disediakan oleh .NET *Framework* untuk menunjang dan mempermudah pengembangan aplikasi yang dibuat diatas .NET *Framework*. Kelas XmlDocument yang berada didalam *namespace* System.Xml berisi berbagai *method* dan properti yang digunakan untuk merepresentasikan sebuah dokumen XML. Sebagai contoh *method* Load() yang digunakan untuk membuka sebuah dokumen XML dan *method* Save() yang digunakan untuk menyimpan perubahan yang dilakukan pada berkas XML yang dibuka sebelumnya [7].

Sedangkan, kelas XmlSerializer yang berada didalam *namespace* System.Xml.Serialization berisi berbagai *method* dan properti untuk melakukan serialisasi atau deserialisasi sebuah objek kedalam bentuk dokumen XML. Sebagai contoh *method* Serialize() yang digunakan untuk melakukan serialisasi dan Deserialize() untuk sebaliknya. Dalam tugas akhir ini, digunakan kedua *class library* tersebut untuk melakukan pengelolaan berkas XML dan membuat berkas XML baru [8].

2.9. Javascript

Javascript adalah bahasa pemrograman dinamis yang biasanya digunakan pada halaman *web*. Javascipt mengimplementasi *client-side scripts* untuk berinteraksi dengan pengguna, melakukan komunikasi secara asynchronous, atau mengubah isi dari dokumen *web* yang ditampilkan.

Dalam tugas akhir ini, digunakan salah satu fitur Javascript yaitu Javascript *Timing Events*. Terdapat 2 fungsi dari Javascript Timing Events yaitu setInterval() dan setTimeOut(), namun yang digunakan pada tugas akhir ini hanya fungsi setTimeOut(). Keduanya bertujuan untuk mengeksekusi fungsi setelah batas waktu yang ditentukan, perbedaannya pada fungsi setTimeOut fungsi tersebut hanya dijalankan sekali saja sedangkan pada setInterval() fungsi tersebut dijalankan secara berulang selama interval waktu yang ditentukan [9].

2.10. jQuery

jQuery merupakan pustaka Javascript gratis dan *open-source* yang didesain untuk memudahkan *client-side scripting* pada halaman *web*. jQuery merupakan pustaka javascript yang paling populer digunakan saat ini. Sintaks jQuery mempermudah navigasi dokumen, memilih atau menyeleksi DOM (Document Object Model) *Elements*, membuat animasi, propagasi *event*, dan mengembangkan aplikasi dengan AJAX [10].

2.11. JSON (*Javascript Object Notation*)

JSON merupakan salah satu format pertukaran data dengan keunggulannya yang ringan serta mudah dalam membaca dan menulisnya. JSON dibangun dengan 2 struktur:

1. Koleksi dari pasangan nilai/nama.
2. Nilai-nilai yang terurut dalam sebuah *list* [11].

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan tahapan analisis dan perancangan tugas akhir. Analisis meliputi kebutuhan-kebutuhan yang diperlukan oleh perangkat lunak. Dari hasil analisis yang telah dilakukan, kemudian dilakukan tahapan perancangan sistem. Perancangan perangkat lunak direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

Dalam tahapan analisis, penulis membagi ke dalam beberapa tahapan antara lain analisis permasalahan, analisis proses CodeReplayer, deskripsi umum sistem, dan kebutuhan perangkat lunak.

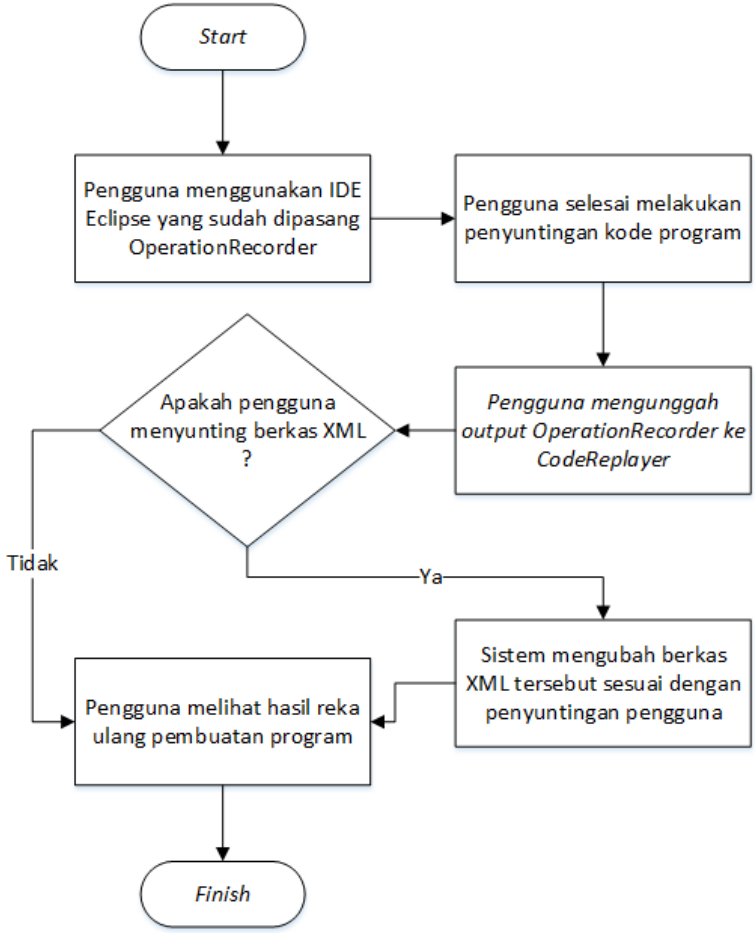
3.1.1. Analisis Permasalahan

Metode umum yang biasa digunakan ketika belajar pemrograman adalah dengan membaca kode sumber atau implementasi dari program yang dipelajari. Namun, hal tersebut memiliki kelemahan karena kode sumber bersifat statis sedangkan penulisan kode sumber adalah serangkaian proses dinamis yang cenderung ditulis tidak berurutan dari baris satu ke baris selanjutnya. Maka dari itu, dibutuhkan solusi yang dapat mempertemukan antara subjek dan kode sumber dengan cara yang lebih baik.

Dengan adanya permasalahan tersebut, dalam tugas akhir ini dibuat sebuah aplikasi berbasis *website* untuk memudahkan proses memahami kode program, CodeReplayer. Aplikasi ini memanfaatkan *plug-in* OperationRecorder, yang terintegrasi dengan IDE (*Integrated Development Environment*) Eclipse dan berfungsi untuk melakukan perekaman operasi secara kronologis dan otomatis ketika seseorang menulis kode program.

Plug-in OperationRecorder akan menghasilkan sebuah bekas dengan format XML yang kemudian akan menjadi *input* pada

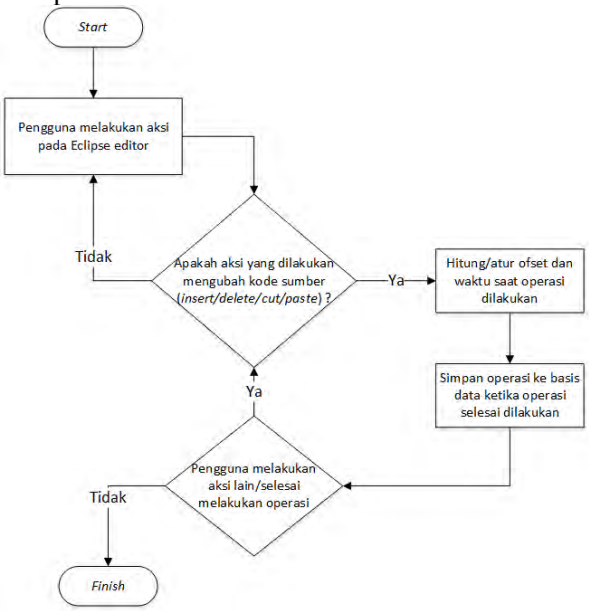
CodeReplayer. Selanjutnya, CodeReplayer akan mengolah berkas XML tersebut untuk kemudian melakukan reka ulang proses produksi kode program runut secara kronologis. CodeReplayer juga dapat melakukan proses pengeditan waktu jeda mengingat perilaku dari *plug-in* OperationRecorder yang melakukan perekaman secara otomatis. Secara keseluruhan, diagram alir yang menggambarkan proses sistem ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Sistem

3.1.2. Analisis *plug-in* OperationRecorder

OperationRecorder merekam setiap operasi yang menyebabkan berubahnya kode program pada editor Eclipse. Operasi yang direkam seperti ketik/tambah, hapus, modifikasi, hingga *cut* dan *paste* karakter. OperationRecorder tidak merekam operasi yang tidak melakukan pengubahan pada kode program secara langsung seperti perpindahan kursor, seleksi teks, membuka dan menutup berkas, berpindah berkas, mengubah nama, dan melakukan *undo*/*redo*. Diagram alir OperationRecorder ditunjukkan pada Gambar 3.2.

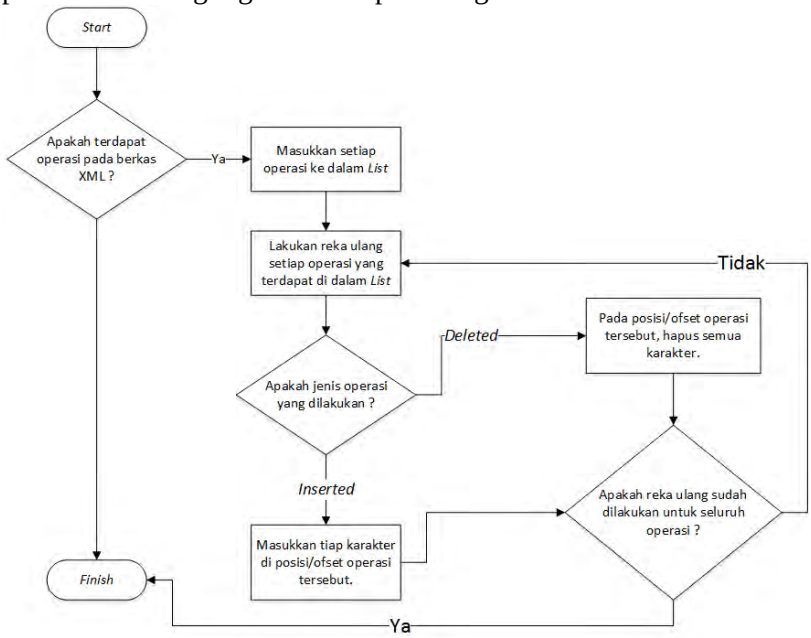


Gambar 3.2 Diagram alir OperationRecorder

3.1.3. Analisis proses pada CodeReplayer

Seperti yang telah dijelaskan sebelumnya, bahwa CodeReplayer akan mengolah berkas XML yang merupakan keluaran dari *plug-in* OperationRecorder. Setelah berkas tersebut diolah sistem, akan dilakukan proses reka ulang pembuatan kode

program. Proses dari berkas XML diunggah hingga dihasilkan proses reka ulang digambarkan pada diagram alir Gambar 3.3.



Gambar 3.3 Diagram alir CodeReplayer

Secara garis besar, proses dalam CodeReplayer ini akan dibagi dalam dua tahap yaitu proses pengolahan berkas XML dan proses reka ulang yang akan dijelaskan secara lebih detil sebagai berikut.

3.1.3.1. Analisis Proses Pengolahan Berkas XML

Pada proses ini, *input* berupa berkas XML hasil dari perekaman *plug-in* OperationRecorder. Maka, pada aplikasi CodeReplayer ini akan diberikan sebuah fitur untuk melakukan unggah berkas XML. Dari berkas XML yang diunggah tersebut, kemudian sistem akan memproses agar tiap *node* pada berkas XML tersebut dapat dibaca dan diproses untuk melakukan proses selanjutnya yaitu reka ulang.

3.1.3.2. Analisis Proses Reka Ulang

Ketika sistem sudah dapat mengolah berkas XML yang merupakan *output* dari *plug-in* OperationRecorder sesuai dengan hasil yang diharapkan, maka tahap selanjutnya adalah melakukan reka ulang untuk tiap *node* pada berkas XML tersebut. Adapun, proses reka ulang ini membutuhkan *input* berupa hasil pengolahan berkas XML dari proses sebelumnya. Proses reka ulang ini dibuat menggunakan bahasa Javascript dan menggunakan salah satu fungsi pada pustaka Javascript *Timing Events*, yaitu *setTimeOut()*.

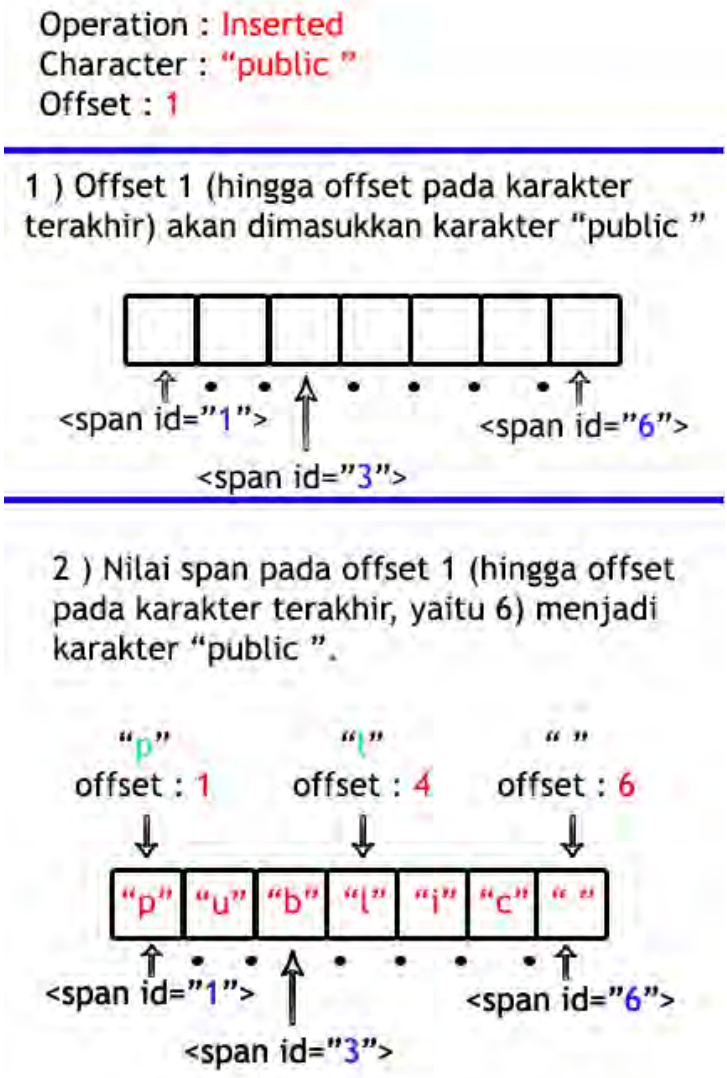
Pada berkas XML OperationRecorder terdapat nilai ofset yang menjadi posisi di mana sebuah karakter akan dioperasikan. Dari atribut ofset, sistem dapat menempatkan tiap karakter tersebut ke dalam wadah yang merupakan salah satu elemen HTML yaitu *span*. Sistem akan membuat *span* sebanyak jumlah karakter yang akan dioperasikan dan tiap-tiap *span* akan mewakili 1 karakter sesuai dengan ofset dari masing-masing karakter tersebut.

Jika operasi yang dilakukan adalah *inserted*, maka sistem akan menempatkan setiap karakter pada operasi tersebut di ofset yang tertulis pada operasi tersebut. Ilustrasi operasi *inserted* ditunjukkan pada Gambar 3.4.

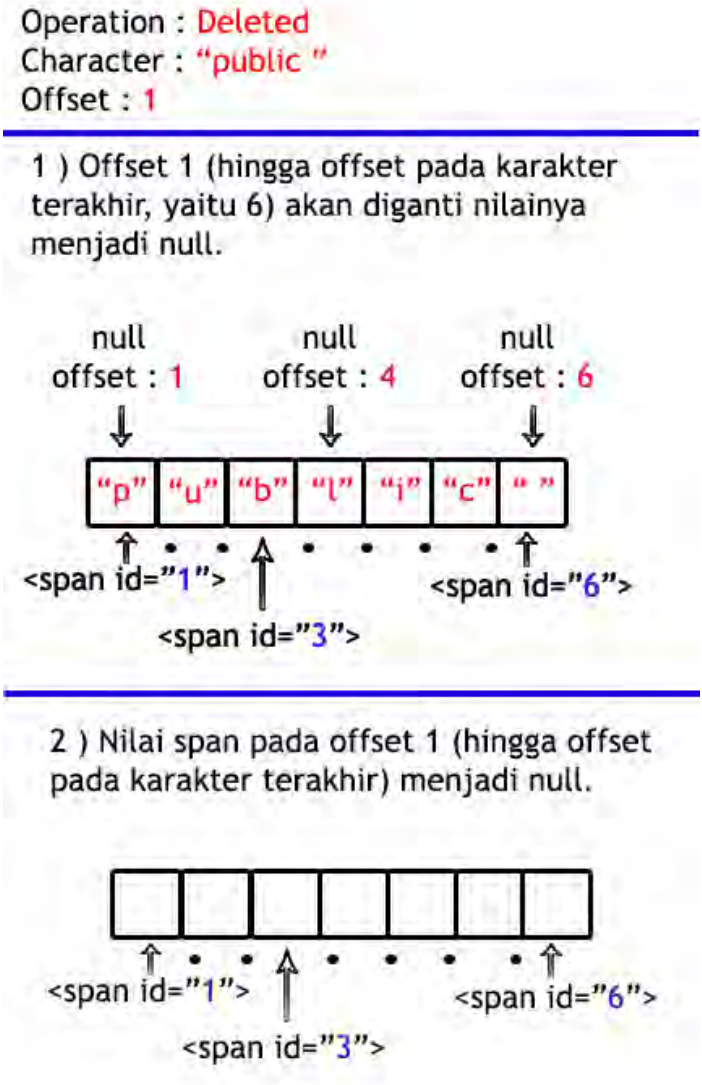
Sedangkan jika operasi yang dilakukan adalah *deleted*, maka sistem akan mengubah setiap karakter mulai dari ofset operasi tersebut hingga akhir karakter menjadi bernilai *null*. Ilustrasi operasi *deleted* ditunjukkan pada Gambar 3.5.

3.1.4. Deskripsi Umum Perangkat Lunak

Perangkat lunak yang dibangun adalah aplikasi berbasis *web* yang dibangun menggunakan kerangka kerja ASP.NET MVC 5 dan Entity *Framework* 6. Aplikasi berfungsi untuk melakukan reka ulang proses pembuatan kode program yang sebelumnya direkam oleh *plug-in* OperationRecorder. *Output* dari *plug-in* OperationRecorder berupa berkas XML akan menjadi *input* dari aplikasi ini untuk diolah dan dilakukan reka ulang proses pembuatan kode program.



Gambar 3.4 Ilustrasi proses reka ulang operasi *inserted*



Gambar 3.5 Ilustrasi proses reka ulang operasi *deleted*

3.1.5. Spesifikasi Kebutuhan Perangkat Lunak

Dalam subbab ini membahas spesifikasi kebutuhan fungsional sistem dari hasil analisis penulis. Bagian ini berisi semua kebutuhan perangkat lunak yang ditulis dalam bentuk kebutuhan fungsional, diagram kasus penggunaan, skenario kasus penggunaan, diagram aktivitas, dan diagram sekuens.

3.1.5.1. Kebutuhan Fungsional

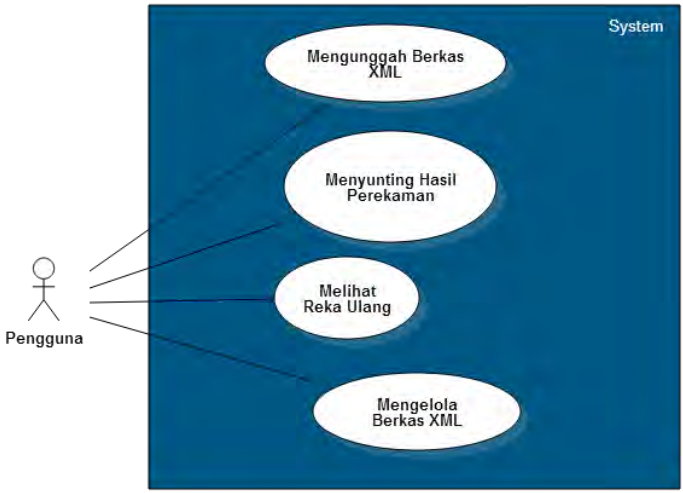
Kebutuhan fungsional merupakan proses-proses yang harus ada pada sistem sehingga sistem dapat berjalan dengan baik, fungsi yang merupakan kebutuhan utama dari sistem. Kebutuhan fungsional mendefinisikan layanan yang harus dimiliki oleh sistem, bagaimana reaksi sistem terhadap masukan yang ada dan apa yang dilakukan sistem pada situasi khusus. Adapun kebutuhan fungsional CodeReplayer ini dijelaskan pada Tabel 3.1.

Tabel 3.1 Kebutuhan Fungsional

Kode Kebutuhan Fungsional	Kebutuhan Fungsional	Deskripsi
FR01	Manajemen Berkas XML	Digunakan untuk melihat daftar berkas XML yang pernah diunggah oleh pengguna termasuk didalamnya melakukan unggah berkas baru atau menghapusnya.
FR02	Menyunting Berkas XML	Digunakan untuk melakukan sunting dari berkas XML yang telah diunggah, penyuntingan dapat berupa mengurangi waktu jeda atau menambahkan catatan dokumentasi.
FR03	Menampilkan Reka Ulang Proses Pembuatan Program	Digunakan untuk melihat hasil reka ulang dari berkas XML yang telah diunggah dan/atau disunting sebelumnya.

3.1.5.2. Diagram Kasus Penggunaan

Kasus penggunaan sistem berdasarkan hasil analisis kebutuhan fungsional sistem dijelaskan secara rinci pada subbab ini. Ada 4 kasus penggunaan pada sistem ini sebagaimana yang ada pada Gambar 3.6. Penjelasan kasus penggunaan sistem lebih rinci dapat dilihat pada Tabel 3.2.



Gambar 3.6 Diagram Kasus Penggunaan Aktor Pengguna

Tabel 3.2 Daftar Kasus Penggunaan

Kode Kasus Pengunaan	Nama
UC01	Mengunggah Berkas XML
UC02	Menyunting Hasil Perekaman
UC03	Melihat Reka Ulang
UC04	Mengelola Berkas XML

3.1.5.3. Skenario Kasus Penggunaan

a) Mengunggah Berkas XML

Pada kasus penggunaan ini, sistem akan menerima masukan berupa berkas XML yang merupakan hasil dari plug-in

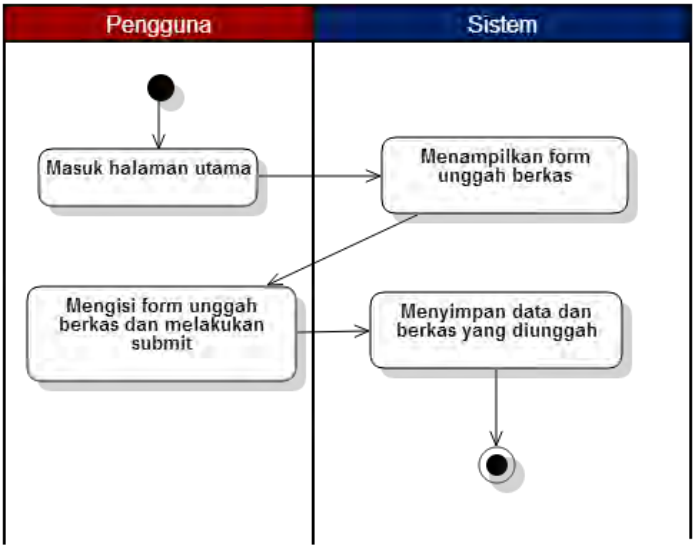
OperationRecorder. Hasil akhir dari kasus penggunaan ini, sistem akan menyimpan berkas XML pada *server* dan menyimpan data terkait berkas tersebut pada basis data.

Tabel 3.3 Skenario Mengunggah Berkas XML

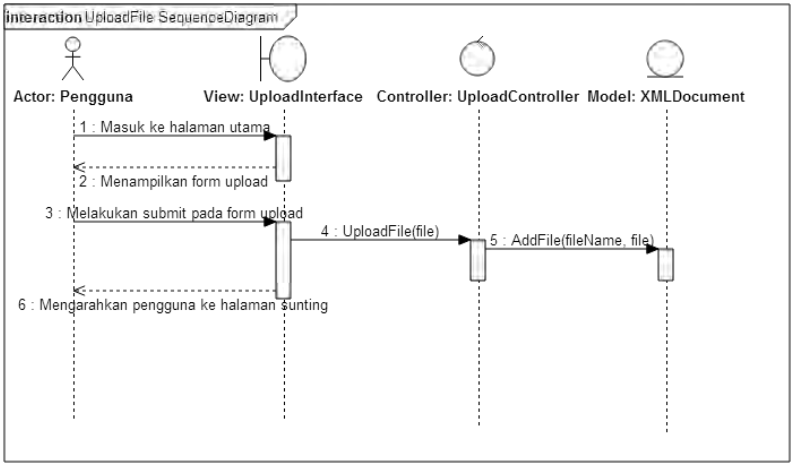
Nama Kasus Penggunaan	Mengunggah Berkas XML
Nomor	UC01
Deskripsi	Pengguna mengunggah berkas XML hasil perekaman *plug-in* OperationRecorder
Aktor	Pengguna
Kondisi Awal	Pengguna telah melakukan *login*
Alur Normal	1. Pengguna masuk ke halaman utama/beranda.
	2. Pengguna mengisi nama/judul dari berkas XML yang diunggah
	3. Pengguna memilih dan mengunggah berkas XML yang diinginkan
Alur Alternatif	-
Kondisi Akhir	Ditambahkan *record* baru berkas XML yang diunggah pada akun pengguna yang bersangkutan

Berdasarkan skenario kasus penggunaan mengunggah berkas XML yang ada pada Tabel 3.3, selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Untuk diagram aktivitas dari kasus penggunaan mengunggah berkas XML dapat dilihat pada Gambar 3.7. Dari diagram aktivitas yang ada pada Gambar 3.7, selanjutnya dibentuk diagram sekuens yang dapat dilihat pada Gambar 3.8.

Pada kasus penggunaan ini, pengguna memiliki hak akses untuk mengunggah berkas XML yang merupakan hasil dari *plug-in* OperationRecorder. Setelah pengguna melengkapi form unggah dan menekan tombol Submit, maka sistem akan menyimpan berkas XML yang diunggah di *server* dan data berkas tersebut pada basis data sistem.



Gambar 3.7 Diagram Aktivitas Mengunggah Berkas XML



Gambar 3.8 Diagram Sekuens Mengunggah Berkas XML

b) Menyunting Hasil Perekaman

Pada kasus penggunaan menyunting hasil perekaman, sistem akan membaca berkas XML yang dipilih untuk disunting. Berkas XML ini dipilih dari data berkas XML yang sebelumnya telah diunggah oleh pengguna. Atribut yang dapat diubah berupa waktu jeda tiap operasi dan kecepatan pengetikan tiap karakter. Selain itu, dapat pula ditambahkan catatan atau dokumentasi dari operasi yang dilakukan untuk mempermudah dalam memahami kode program pada operasi tersebut.

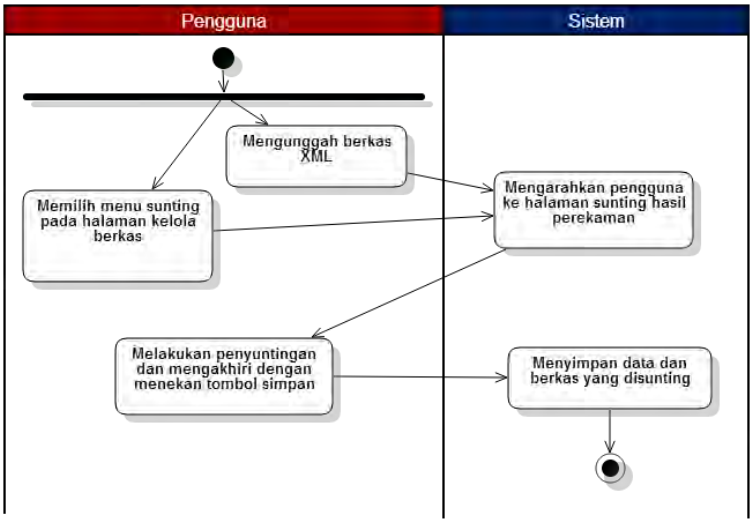
Hasil dari kasus penggunaan ini berupa berkas XML baru yang disimpan pada *server* dan merupakan salinan dari berkas XML yang disunting dengan atribut-atribut yang telah diubah.

Berdasarkan skenario kasus penggunaan menyunting hasil perekaman yang ada pada Tabel 3.4, selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Untuk diagram aktivitas dari kasus penggunaan menyunting hasil perekaman dapat dilihat pada Gambar 3.9. Dari diagram aktivitas yang ada pada Gambar 3.9, selanjutnya dibentuk diagram sekuens yang dapat dilihat pada Gambar 3.10.

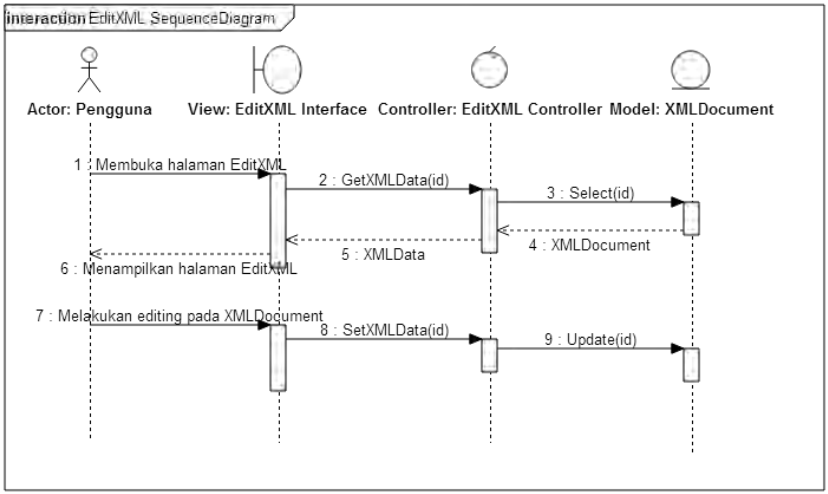
Pada kasus penggunaan ini, pengguna memiliki hak akses untuk menyunting atau melakukan pengubahan data pada berkas XML yang sebelumnya telah diunggah. Setelah pengguna selesai melakukan penyuntingan dan menekan tombol simpan, maka sistem akan menyimpan hasil berkas XML yang telah disunting di server dan data berkas tersebut pada basis data sistem.

Tabel 3.4 Skenario Menyunting Hasil Perekaman

Nama Kasus Penggunaan	Menyunting Hasil Perekaman
Nomor	UC02
Deskripsi	Pengguna menyunting hasil dari perekaman *plug-in* OperationRecorder
Aktor	Pengguna
Kondisi Awal	Pengguna telah mengunggah berkas XML yang ingin disunting
Alur Normal	1. Pengguna mengunggah berkas XML A1. Pengguna masuk ke halaman kelola berkas XML
	2. Pengguna diarahkan sistem ke halaman sunting hasil perekaman
	3. Pengguna melakukan penyuntingan dan menekan tombol simpan
Alur Alternatif	A1. Pengguna masuk ke halaman kelola berkas XML A1.1. Pengguna memilih menu sunting pada berkas XML yang ingin disunting. A1.2. Pengguna diarahkan sistem ke halaman sunting hasil perekaman
Kondisi Akhir	Disimpan atribut-atribut dari berkas XML yang telah disunting beserta berkas XML yang telah disunting



Gambar 3.9 Diagram Aktivitas Menyunting Hasil Perekaman



Gambar 3.10 Diagram Sekuens Menyunting Hasil Perekaman

c) Melihat Reka Ulang

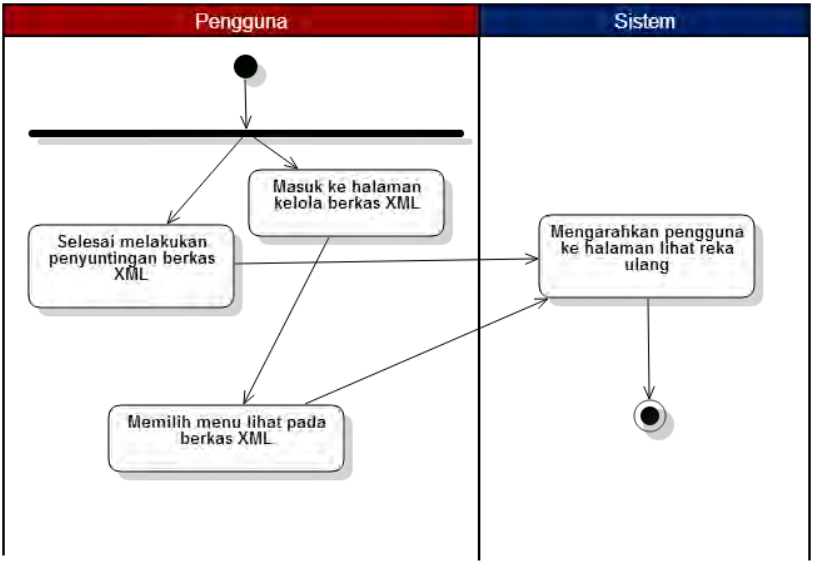
Pada kasus penggunaan melihat reka ulang, sistem akan membaca berkas XML yang sebelumnya diunggah atau dipilih oleh pengguna. Hasil dari kasus penggunaan ini adalah skenario reka ulang proses pembuatan kode program yang sebelumnya direkam dengan *plug-in* OperationRecorder.

Tabel 3.5 Skenario Melihat Reka Ulang

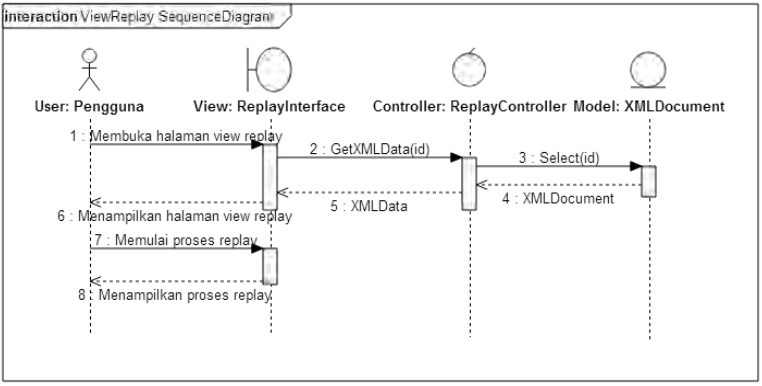
Nama Kasus Penggunaan	Melihat Reka Ulang
Nomor	UC03
Deskripsi	Pengguna melihat skenario reka ulang pembuatan kode program
Aktor	Pengguna
Kondisi Awal	Pengguna telah mengunggah berkas XML
Alur Normal	1. Pengguna selesai melakukan penyuntingan A1. Pengguna masuk ke halaman kelola berkas XML
	2. Pengguna diarahkan sistem ke halaman lihat reka ulang
Alur Alternatif	A1. Pengguna masuk ke halaman kelola berkas XML A1.1. Pengguna memilih menu lihat pada berkas XML yang ingin dilihat reka ulangnya. A1.2. Pengguna diarahkan sistem ke halaman lihat reka ulang.
Kondisi Akhir	Pengguna melihat hasil reka ulang dari perekaman *plug-in* OperationRecorder

Berdasarkan skenario kasus penggunaan melihat reka ulang yang ada pada Tabel 3.5, selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Untuk diagram aktivitas dari kasus penggunaan melihat reka ulang dapat dilihat pada Gambar 3.11. Dari diagram aktivitas yang ada pada Gambar 3.11, dibentuk diagram sekuens yang dapat dilihat pada Gambar 3.12.

Pada kasus penggunaan ini, pengguna dapat melakukan proses reka ulang pada berkas XML yang sebelumnya telah diunggah dan disimpan datanya pada *server*.



Gambar 3.11 Diagram Aktivitas Melihat Reka Ulang



Gambar 3.12 Diagram Sekuens Melihat Reka Ulang

d) Mengelola Berkas XML

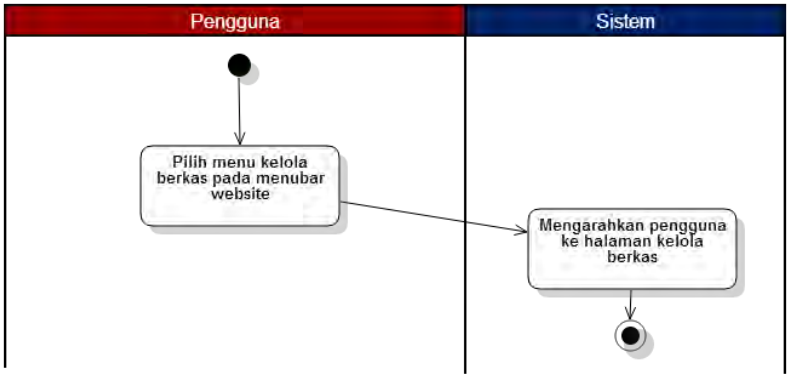
Pada kasus penggunaan mengelola berkas XML, sistem akan membaca data berkas XML yang pernah diunggah oleh pengguna. Hasil atau keluaran dari kasus penggunaan ini berupa tabel yang berisi data atau properti berkas XML yang sebelumnya telah diunggah.

Tabel 3.6 Skenario Mengelola Berkas XML

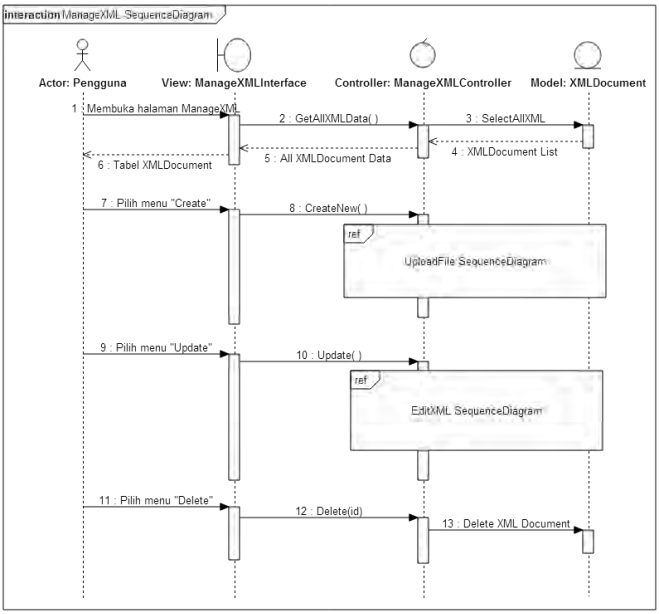
Nama Kasus Penggunaan	Mengelola Berkas XML
Nomor	UC04
Deskripsi	Pengguna melakukan pengelolaan pada berkas XML yang pernah diunggahnya ke sistem
Aktor	Pengguna
Kondisi Awal	Pengguna telah mengunggah berkas XML yang ingin dikelola
Alur Normal	1. Pengguna memilih menu kelola berkas
	2. Pengguna diarahkan sistem ke halaman kelola berkas
Alur Alternatif	-
Kondisi Akhir	Diperbarui data atribut pada berkas XML jika pengguna melakukan pembaruan pada data tersebut

Berdasarkan skenario kasus penggunaan mengelola berkas XML yang ada pada Tabel 3.6, selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Untuk diagram aktivitas dari kasus penggunaan mengelola berkas XML dapat dilihat pada Gambar 3.13. Dari diagram aktivitas yang ada pada Gambar 3.13, selanjutnya dibentuk diagram sekuens yang dapat dilihat pada Gambar 3.14.

Pada kasus penggunaan ini, pengguna dapat melihat data dari berkas XML yang telah diunggah sebelumnya. Data ini berupa tabel yang memuat detil dari tiap berkas XML. Dari halaman ini pula, pengguna dapat melakukan tambah, sunting, atau bahkan menghapus berkas XML yang telah diunggah.



Gambar 3.13 Diagram Aktivitas Mengelola Berkas XML



Gambar 3.14 Diagram Sekuens Mengelola Berkas XML

3.2. Perancangan Sistem

Pada subbab ini dijelaskan mengenai tahapan perancangan sistem. Perancangan sistem ini dibagi menjadi empat bagian, yaitu perancangan basis data, perancangan arsitektur perangkat lunak, dan perancangan antarmuka pengguna.

3.2.1. Perancangan Basis Data

Perancangan basis data merupakan perancangan tabel-tabel yang dibutuhkan dalam proses pembangunan sistem beserta penggambaran hubungan antar tabel.

3.2.1.1. Rancangan Tabel Pengguna

Tabel ini digunakan untuk menyimpan data anggota dari sebuah tim prestasi. Penjelasan mengenai tabel anggota dapat dilihat pada Tabel 3.7.

Tabel 3.7 Atribut Tabel Pengguna

Nama Kolom	Keterangan
Username	Merupakan *primary key* dari tabel pengguna
Password	*Password* pengguna untuk *login* ke sistem.

3.2.1.2. Rancangan Tabel Berkas XML

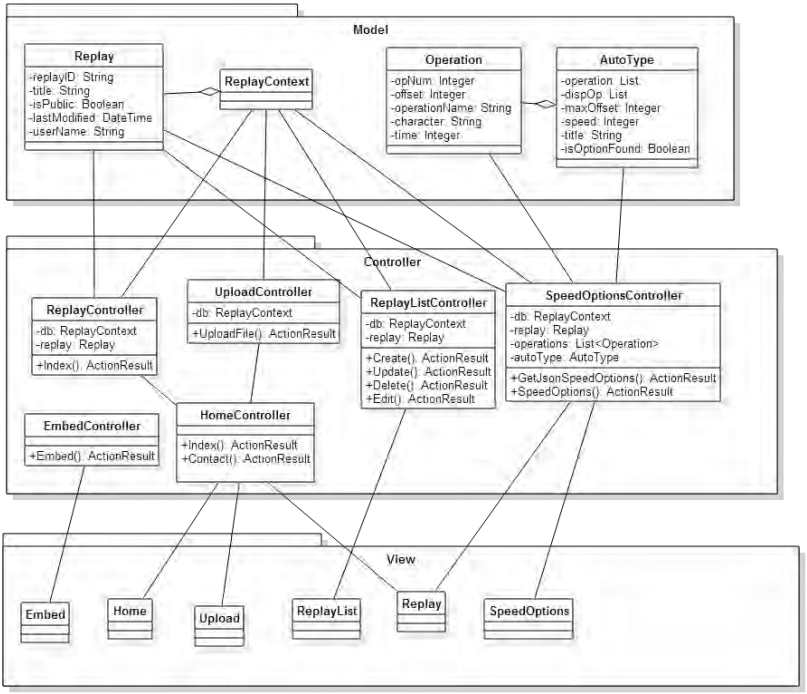
Tabel ini digunakan untuk menyimpan berita. Penjelasan mengenai tabel berita dapat dilihat pada Tabel 3.8.

Tabel 3.8 Atribut Tabel Berkas XML

Nama Kolom	Keterangan
ID	Merupakan *primary key*
Username	Merupakan *foreign key* dari tabel pengguna
Title	Judul dari berkas XML
IsPublic	Hak akses berkas XML
LastModified	Waktu terakhir berkas XML disunting

3.2.2. Perancangan Kelas

Perancangan diagram kelas merupakan perancangan kelas-kelas yang dibutuhkan dalam proses pembangunan sistem beserta penggambaran hubungan antar kelas. Adapun diagram kelas pada sistem CodeReplayer ditunjukkan pada Gambar 3.15.



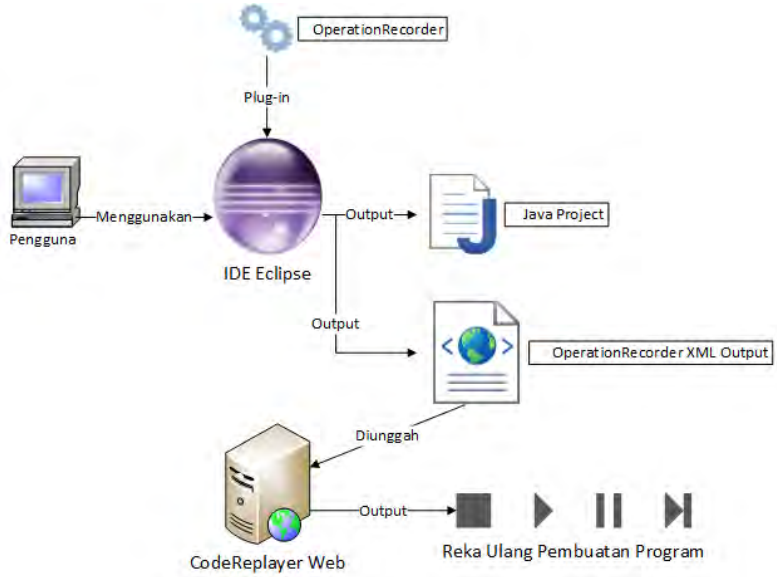
Gambar 3.15 Diagram Kelas

3.2.3. Perancangan Arsitektur Sistem

Rancangan arsitektur CodeReplayer dapat dilihat pada Gambar 3.16.

3.2.4. Perancangan Antarmuka Grafis

Pada subbab ini akan dijelaskan mengenai rancangan antarmuka grafis sistem. Rancangan yang dibahas meliputi beberapa antarmuka seperti yang dijelaskan pada Tabel 3.9



Gambar 3.16 Rancangan Arsitektur CodeReplayer

Tabel 3.9 Rancangan Antarmuka

No	Rancangan Antarmuka
1.	Home
2.	Replay
3.	ReplayList
4.	SpeedOptions
5.	Upload

3.2.4.1. Home

Pada rancangan antarmuka Home akan memuat *form* untuk pengguna *login* ke dalam sistem. Gambar rancangan antarmuka Home ditunjukkan pada Gambar 3.17 dan spesifikasi atribut pada antarmuka Home ditunjukkan pada Tabel 3.10.

Gambar 3.17 Rancangan Antarmuka Home

Tabel 3.10 Spesifikasi Antarmuka Home

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1.	Username Label	Label	Menampilkan teks Username untuk menunjukkan *field* yang akan diisi.
2.	Username Input	Textbox	Menyediakan *field* yang menerima *input* berupa *username*.
3.	Password Label	Label	Menampilkan teks Password untuk menunjukkan *field* yang akan diisi.
4.	Password Input	Textbox	Menyediakan *field* yang menerima *input* berupa *password*.
5.	Remember Account Checkbox	Checkbox	Memberikan pilihan jika pengguna ingin menyimpan data akunnya pada *cache browser* pengguna.
6.	Button Login	Button	Tombol yang dapat diklik oleh pengguna untuk masuk ke dalam sistem setelah mengisi data akun pengguna.
7.	Register Link	Hyperlink	Hyperlink yang dapat diklik oleh pengguna jika pengguna belum terdaftar pada sistem dan ingin melakukan registrasi.

3.2.4.2. Replay

Pada rancangan antarmuka Replay akan memuat *canvas* untuk pengguna melihat hasil reka ulang dari berkas XML yang diunggah serta beberapa *button* yang digunakan untuk mengontrol proses reka ulang ditambah *sidebox* untuk memberikan penjelasan terkait proses reka ulang yang sedang dilakukan. Gambar rancangan

antarmuka Replay ditunjukkan pada Gambar 3.18 dan spesifikasi atribut pada antarmuka Replay ditunjukkan pada Tabel 3.11.

Gambar 3.18 Rancangan Antarmuka Replay

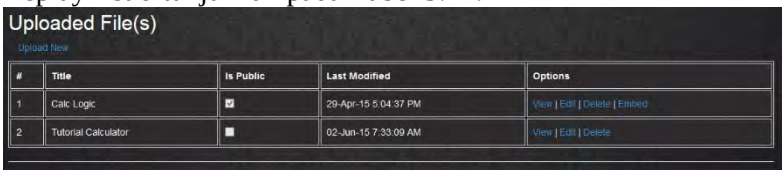
Tabel 3.11 Spesifikasi Antarmuka Replay

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1.	Replay Canvas	Canvas	Bagian atas canvas memuat judul dari proses reka ulang yang dilakukan dan bagian bawah yang berwarna gelap memuat hasil reka ulang.
2.	Button Kontrol Replay	Button	Beberapa button digunakan untuk melakukan kontrol reka ulang, seperti *button stop*, *play*, *pause*, dan *fast forward*.
3.	Note Canvas	Canvas	Menampilkan teks yang berisi penjelasan dari proses reka ulang yang sedang berjalan.

3.2.4.3. ReplayList

Pada rancangan antarmuka ReplayList akan memuat tabel untuk menampilkan berkas XML yang telah diunggah oleh pengguna. Tabel tersebut memuat beberapa atribut seperti: judul berkas, hak akses berkas, tanggal modifikasi terakhir, dan opsi untuk melihat berkas, mengedit, menghapus, dan melakukan *embed*. Gambar rancangan antarmuka ReplayList ditunjukkan

pada Gambar 3.19 dan spesifikasi atribut pada antarmuka ReplayList ditunjukkan pada Tabel 3.12.



Gambar 3.19 Rancangan Antarmuka ReplayList

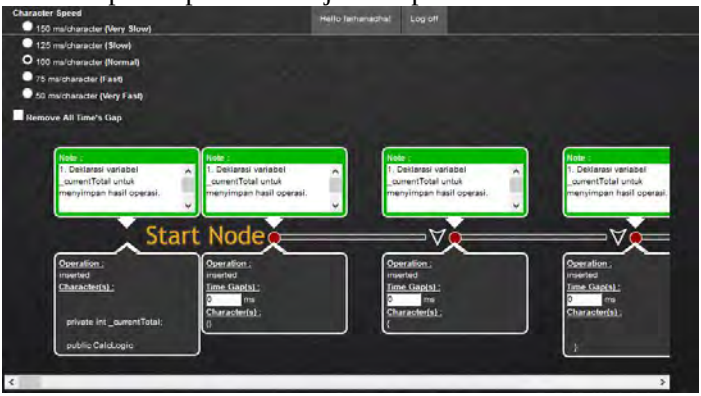
Tabel 3.12 Spesifikasi Antarmuka ReplayList

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1.	Upload New Link	Hyperlink	Hyperlink yang dapat diklik oleh pengguna untuk mengunggah berkas XML yang baru.
2.	Title Atribut	Label	Menunjukkan judul kolom yang berisi data judul dari berkas XML.
3.	IsPublic Atribut	Label	Menunjukkan judul kolom yang berisi hak akses dari berkas XML.
4.	LastModified Atribut	Label	Menunjukkan judul kolom yang berisi tanggal terakhir dari penyuntingan berkas XML.
5.	Options Atribut	Label	Menunjukkan judul kolom yang berisi pilihan menu yang dapat dilakukan pada berkas XML.
6.	IsPublic checkbox	Checkbox	Menampilkan data apakah berkas XML yang diunggah bersifat publik atau tidak.
7.	Options Menu	Hyperlink	Hyperlink yang dapat diklik oleh pengguna untuk mengelola berkas XML.

3.2.4.4. SpeedOptions

Pada rancangan antarmuka SpeedOptions akan memuat pilihan dari kecepatan karakter yang dapat dipilih pengguna dan panel penyuntingan tiap *node* dari berkas XML yang diunggah. Selain itu, sebagai tambahan, disediakan pula kolom isian Note yang dapat digunakan untuk memberikan penjelasan tambahan terkait

node tersebut. Gambar rancangan antarmuka SpeedOptions ditunjukkan pada Gambar 3.20 dan spesifikasi atribut pada antarmuka SpeedOptions ditunjukkan pada Tabel 3.13.



Gambar 3.20 Rancangan Antarmuka SpeedOptions

Tabel 3.13 Spesifikasi Antarmuka SpeedOptions

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1.	Character Speed Label	Label	Menampilkan teks yang menunjukkan pilihan kecepatan yang dapat dipilih pengguna.
2.	Speed Options	Options	Pilihan kecepatan karakter yang dapat dipilih pengguna.
3.	Remove Gap Checkbox	Checkbox	Checkbox yang jika dipilih akan menghapus seluruh jeda waktu antar *node* pada berkas XML.
4.	Note Textbox	Textbox	Textbox tempat pengguna dapat mengisi penjelasan tambahan terkait *node* tersebut.
5.	Node	Text	Memberikan penjelasan terkait tiap *node* yang ada pada berkas XML yang diunggah.

3.2.4.5. Upload

Pada rancangan antarmuka Upload akan memuat *form* untuk pengguna mengunggah berkas ke dalam sistem. Gambar rancangan

antarmuka Upload ditunjukkan pada Gambar 3.21 dan spesifikasi atribut pada antarmuka Upload ditunjukkan pada Tabel 3.14.

Gambar 3.21 Rancangan Antarmuka Upload

Tabel 3.14 Spesifikasi Antarmuka Upload

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan
1.	Title Label	Label	Menampilkan label yang menunjukkan *field* yang akan diisi adalah judul berkas.
2.	Title Input	Textbox	Menyediakan *field* yang menerima *input* berupa judul berkas.
3.	Browse	Button	Button untuk memilih berkas XML yang akan diunggah pada perangkat pengguna.
4.	Drop Area	Div	Menyediakan area yang dapat menerima *input* berupa berkas XML yang dipilih pengguna dengan melakukan *drag* ke Drop Area.

BAB IV
IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi pembangunan sistem berdasarkan analisis dan perancangan yang telah dijelaskan pada bab sebelumnya. Implementasi ini meliputi implementasi kelas pada lapisan data, kontrol, dan antarmuka.

4.1. Lingkungan Implementasi

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah sebuah *laptop*. Spesifikasi dari perangkat tersebut adalah Intel(R) Core (TM) i5-4200M CPU @2.50GHz 2.50GHz, RAM 8.00 GB.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. Microsoft Windows 8.164-bit sebagai sistem operasi.
2. Microsoft Visual Studio 2013 sebagai tools untuk mengimplementasikan aplikasi.
3. Eclipse Java EE 4.4.1 sebagai kakas bantu untuk menguji *plug-in* OperationRecorder.
4. *Plug-in* OperationRecorder sebagai kakas bantu untuk melakukan perekaman proses pembuatan kode program pada Eclipse.
5. Internet Information Services (IIS) sebagai *web server*.

4.2. Implementasi Aplikasi

4.2.1. Implementasi Lapisan Model

Lapisan model merupakan lapisan yang terdiri dari kelas-kelas yang bersifat sebagai penyimpanan. Lapisan ini terdiri dari atribut-atribut yang digunakan untuk menyimpan elemen-elemen pada saat membangun sistem. Lapisan ini hanya dapat diakses oleh

lapisan kontrol. Dalam aplikasi CodeReplayer ini, terdapat 6 kelas model yang ditampilkan pada Tabel 4.1.

Tabel 4.1 Daftar Kelas Model

No	Nama Kelas Model
1.	AccountViewModel
2.	AutoType
3.	IdentityModels
4.	Operation
5.	Replay
6.	ReplayContext

4.2.1.1. Kelas AccountViewModel

Kelas AccountViewModel merupakan kelas yang mendefinisikan atribut-atribut terkait aktivitas *login* dan register pengguna pada sistem.

4.2.1.2. Kelas AutoType

Kelas AutoType adalah kelas yang mendefinisikan atribut-atribut untuk melakukan reka ulang. Adapun beberapa atribut pada kelas AutoType ditampilkan pada Tabel 4.2.

4.2.1.3. Kelas IdentityModels

Kelas IdentityModels adalah kelas yang mendefinisikan basis data untuk tabel pengguna.

Tabel 4.2 Atribut Kelas AutoType

No	Tipe	Atribut	Keterangan
1	List	Operation	Menampung operasi-operasi untuk melakukan reka ulang.
2	List	DispOp	Menampung operasi-operasi untuk melakukan *preview* saat melakukan penyuntingan.
3	Integer	MaxOffset	Menampung nilai maksimal ofset karakter.
4	Integer	Speed	Menampung kecepatan tiap karakter pada proses reka ulang.
5	String	Title	Menampung judul dari proses reka ulang yang akan dilakukan.
6	Boolean	IsOptionFound	Digunakan untuk melakukan pengecekan apakah sebuah berkas sebelumnya telah disunting.

4.2.1.4. Kelas Operation

Kelas Operation merupakan representasi atribut yang ada pada berkas XML hasil dari *plug-in* OperationRecorder untuk tiap *node*-nya. Adapun beberapa atribut pada kelas Operation ditampilkan pada Tabel 4.3.

Tabel 4.3 Atribut Kelas Operation

No	Tipe	Atribut	Keterangan
1	Integer	OpNum	Operation Number, menyatakan nomor pada tiap *node* terkait.
2	Integer	Time	Waktu terjadinya operasi dalam satuan Integer.
3	Integer	EditingTime	Waktu setelah dilakukan proses penyuntingan.
4	String	OperationName	Nama operasi dari *node* terkait (*insert*/*delete*).
5	Integer	OriginOffset	Nilai ofset *node*.
6	Integer	Offset	Nilai ofset yang diubah pada saat pemrosesan reka ulang.
7	Integer	FirstOffset	Nilai ofset dari karakter pertama tiap *node*.
8	Integer	EndOffset	Nilai ofset dari karakter terakhir tiap *node*.
9	Integer	CurrDeltaOffset	Nilai panjang ofset untuk tiap *node*.
10	String	Character	Karakter yang dioperasikan.
11	Integer	LineNumber	Jumlah baris dari karakter.
12	Note	String	Catatan untuk dokumentasi dari operasi yang dilakukan.
13	IsAnyChar	Boolean	Untuk melakukan pengecekan apakah terdapat karakter selain karakter kosong pada *node* terkait.

4.2.1.5. Kelas Replay

Kelas Replay mendefinisikan atribut yang akan disimpan pada basis data untuk mengelola proses reka ulang dari berkas XML

yang telah diunggah. Adapun beberapa atribut pada kelas Replay ditampilkan pada Tabel 4.4.

Tabel 4.4 Atribut Kelas Replay

No	Tipe	Atribut	Keterangan
1	String	ReplayID	*Primary Key* untuk setiap proses reka ulang
2	String	Title	Judul untuk proses reka ulang.
3	Boolean	IsPublic	Properti untuk mendefinisikan apakah proses reka ulang tersebut *public* atau *private*.
4	DateTime	LastModified	Properti untuk mengetahui waktu terakhir dari proses penyuntingan reka ulang.
5	String	UserName	Merupakan pemilik dari proses reka ulang yang mengunggah berkas XML terkait.

4.2.1.6. Kelas ReplayContext

Kelas ini digunakan untuk mendefinisikan basis data yang akan digunakan untuk kelas Replay yang sebelumnya telah dijelaskan pada subbab sebelumnya.

4.2.2. Implementasi Lapisan Kontrol

Lapisan kontrol merupakan lapisan yang berisi kelas-kelas yang bertugas untuk menghubungkan antara lapisan *model* dan *view*. Dalam aplikasi CodeReplayer ini, terdapat 6 kelas kontrol yang ditampilkan pada Tabel 4.5.

Tabel 4.5 Daftar Kelas Kontrol

No	Nama Kelas Kontrol
1.	AccountController
2.	EmbedController
3.	HomeController
4.	ReplayController
5.	ReplayListController
6.	SpeedOptionsController
7.	UploadController

4.2.2.1. Kelas AccountController

Kelas AccountController merupakan kelas kontrol yang mengatur aktivitas *authentication* pengguna seperti *login*, *logout*, register dan sebagainya.

4.2.2.2. Kelas EmbedController

Kelas EmbedController digunakan untuk mengatur aktivitas yang berhubungan dengan pemasangan *widget* CodeReplayer. *Widget* ini digunakan untuk memasang hasil reka ulang pada halaman *web* lain dengan mencantumkan kode yang diberikan pada halaman Embed. Namun, agar hasil reka ulang dapat di-*embed* di *web* yang diinginkan, pengguna harus memastikan properti berkas XML yang diunggah menjadi publik pada menu kelola berkas.

4.2.2.3. Kelas HomeController

Kelas HomeController melakukan tugas dalam mengarahkan pengguna ke halaman utama dan halaman *contact*. Tiap halaman tersebut direpresentasikan sebagai sebuah ActionResult. Potongan kode program dari kelas HomeController ditunjukkan pada Kode Sumber 4.1.

```
public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Contact Us";

            return View();
        }
    }
```

Kode Sumber 4.1 Kelas HomeController

4.2.2.4. Kelas ReplayController

Kelas ReplayController merupakan kelas kontrol yang bertugas mengarahkan pengguna ke halaman reka ulang yang dipilih pengguna. Potongan dari kode program kelas ReplayController ditunjukkan pada Kode Sumber 4.2.

```
public class ReplayController : Controller
    {
        ReplayContext db = new ReplayContext();
        //
        // GET: /Replay/
        public ActionResult Index(string refNum)
        {
            Replay replay = db.Replays.Find(refNum);
            if (replay == null)
                return HttpNotFound();
            if (replay.UserName !=
User.Identity.GetUserName() && !replay.IsPublic)
                return new
HttpStatusCodeResult(HttpStatusCode.Forbidden);
            return View((object)refNum);
        }
    }
```

Kode Sumber 4.2 Kelas ReplayController

4.2.2.5. Kelas ReplayListController

Kelas ReplayListController adalah kelas kontrol yang digunakan untuk menangani fitur kelola berkas XML. Fitur-fitur dari kelola berkas XML seperti: unggah, sunting, hapus, lihat reka ulang, dan *embed*. Potongan kode program dari kelas ReplayListController ditunjukkan pada Kode Sumber 4.3.

```
[Authorize]
    public class ReplayListController : Controller
    {
        ReplayContext db = new ReplayContext();

        public ActionResult Index()
        {
            string username =
User.Identity.GetUserName();
            var replayList = from b in db.Replays
                             where b.UserName ==
username
                             select b;

            return View(replayList);
        }

        public ActionResult Details(string id)
        {
            return View();
        }

        public ActionResult Create()
        {
            return RedirectToAction("Index",
"Upload");
        }

        [HttpPost]
        public ActionResult Create(FormCollection
collection)
        {
            try
```

```
            {
                // TODO: Add insert logic here

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        public ActionResult Edit(string refNum)
        {
            if (refNum == null)
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            Replay replay = db.Replays.Find(refNum);
            if (replay == null)
                return HttpNotFound();

            return View(replay);
        }

        [HttpPost]
        public ActionResult Edit(Replay replay)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    db.Entry(replay).State =
EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("index");
                }
                return View(replay);
            }
            catch
            {
                return View();
```

```
            }
        }

        public ActionResult Delete(string refNum)
        {
            if (refNum == null)
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            Replay replay = db.Replays.Find(refNum);
            if (replay == null)
                return HttpNotFound();
            return View(replay);
        }

        [HttpPost]
        public ActionResult Delete(string refNum,
Replay rep)
        {
            try
            {
                Replay replay = new Replay();
                if (ModelState.IsValid)
                {
                    if (refNum == null)
                        return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
                    replay = db.Replays.Find(refNum);
                    if (replay == null)
                        return HttpNotFound();
                    db.Replays.Remove(replay);
                    db.SaveChanges();
                    return RedirectToAction("index");
                }
                return View(replay);
            }
            catch
            {
                return View();
            }}}
```

Kode Sumber 4.3 Kelas ReplayListController

4.2.2.6. Kelas SpeedOptionsController

Kelas SpeedOptionsController adalah kelas kontrol yang digunakan untuk melakukan penyuntingan pada berkas XML yang telah diunggah oleh pengguna. Potongan kode program yang mengarahkan pengguna ke halaman sunting berkas XML ditunjukkan pada Kode Sumber 4.4.

```
public ActionResult SpeedOptions(String refNum)
        {
            Replay replay = db.Replays.Find(refNum);
            if (replay == null)
                return HttpNotFound();
            replay.Title = Request["Title"];
            if (Request["IsPublic"] == "false")
                replay.IsPublic = false;
            else
                replay.IsPublic = true;
            replay.LastModified = DateTime.Now;
            XmlDocument doc = new XmlDocument();
            string targetPath =
Server.MapPath("~/Documents/xml-operations/" + refNum
+ ".xml");
            string tempTargetPath =
Server.MapPath("~/Documents/xml-operations/" + refNum
+ "_.xml");
            doc.Load(tempTargetPath);
            XmlNodeList operations =
doc.DocumentElement.SelectNodes("//*[local-
name()='Operation']");
            XmlElement elem;

            elem = doc.CreateElement("Options");
            elem.SetAttribute("title", replay.Title);
            elem.SetAttribute("speed",
Request["speed"]);
            doc.DocumentElement.AppendChild(elem);

            int opLen = operations.Count;
            int num = 0;
            for (int i = 0; i < opLen; i++)
```

```
            {
                if
(operations[i].Attributes["OpNum"].Value ==
num.ToString())
                {
                    if (Request["inputValue" + num]
!= null)

operations[i].Attributes["Time"].Value =
Request["inputValue" + num];

                    elem = doc.CreateElement("Note");
                    elem.InnerText = Request["note" +
num];
                    operations[i].AppendChild(elem);
                    num++;
                }
            }
            doc.PreserveWhitespace = true;
            doc.Save(targetPath);
            System.IO.File.Delete(tempTargetPath);
            db.SaveChanges();
            return RedirectToAction("index",
"replay", new { refNum = refNum });
        }
```

Kode Sumber 4.4 ActionResult SpeedOptions

Pada kelas ini juga terdapat beberapa ActionResult yang melakukan *return* berupa JSON untuk ditransmisikan menuju *view*. Hal ini dimaksudkan untuk mempermudah proses transfer data antar *view* dan kontrol karena data tersebut berukuran besar dan bertipe data *list of list*. Potongan kode program dari ActionResult yang melakukan *return* JSON ditunjukkan pada Kode Sumber 4.5.

```
public ActionResult GetJsonOperations(String refNum)
        {
            AutoType autoType = new AutoType();
            XmlDocument doc = new XmlDocument();
            string targetPath =
Server.MapPath("~/Documents/xml-operations/" + refNum
+ ".xml");
            doc.Load(targetPath);
            XmlNodeList operationsXML =
doc.DocumentElement.SelectNodes("//*[local-
name()='Operation']");
            XmlNode options =
doc.SelectSingleNode("//*[local-name()='Options']");
            List<Operation> operations = new
List<Operation>();
            int maxOffset = 0;
            int opLen = operationsXML.Count;
            string note = null;
            for (int i = 0; i < opLen; i++)
            {Operation op = new Operation();
                op.OpNum =
Convert.ToInt32(operationsXML[i].Attributes["OpNum"].
Value);
                op.Time =
Convert.ToInt64(operationsXML[i].Attributes["Time"].V
alue);
                op.OperationName =
operationsXML[i].Attributes["OperationName"].Value;
                op.OriginOffset =
Convert.ToInt32(operationsXML[i].Attributes["OriginOf
fset"].Value);
                op.Offset =
Convert.ToInt32(operationsXML[i].Attributes["Offset"]
.Value);
                op.FirstOffset =
Convert.ToInt32(operationsXML[i].Attributes["FirstOff
set"].Value);
                op.EndOffset =
Convert.ToInt32(operationsXML[i].Attributes["EndOffse
t"].Value);
```

```
                op.CurrDeltaOffset =
Convert.ToInt32(operationsXML[i].Attributes["CurrDelt
aOffset"].Value);
                op.LineNumber =
Convert.ToInt32(operationsXML[i].Attributes["LineNumb
er"].Value);
                op.IsAnyChar =
Convert.ToBoolean(operationsXML[i].Attributes["IsAnyC
har"].Value);
                op.Character =
operationsXML[i].ChildNodes[0].InnerText;
                XmlNode noteNode =
operationsXML[i].SelectSingleNode("Note");
                if (noteNode != null)
                {
                    note =
operationsXML[i].ChildNodes[1].InnerText.Replace("\n"
, "<br />");
                    op.Note = note;
                }
                else
                    op.Note = note;
                if (op.EndOffset > maxOffset)
                    maxOffset = op.EndOffset;

                operations.Add(op);
            }
            autoType.Operations = operations;
            autoType.MaxOffset = maxOffset;
            autoType.Speed =
Convert.ToInt32(options.Attributes["speed"].Value);
            autoType.Title =
options.Attributes["title"].Value;
            return Json(autoType,
JsonRequestBehavior.AllowGet);
        }
```

Kode Sumber 4.5 ActionResult GetJsonOperations

4.2.2.7. Kelas UploadController

Kelas UploadController merupakan kelas yang menangani dan mengarahkan pengguna pada saat melakukan unggah berkas XML. Potongan dari kode program kelas UploadController ditunjukkan pada Kode Sumber 4.6.

```
public ActionResult UploadFile()
        {
            string path = "";
            long ticks = DateTime.Now.Ticks;

            if (Request.Files["file-
upload"].ContentLength > 0)
            {
                string extension =
System.IO.Path.GetExtension(Request.Files["file-
upload"].FileName);
                path = string.Format("{0}/{1}{2}",
Server.MapPath("~/documents"), ticks, extension);
                if (System.IO.File.Exists(path))
                    System.IO.File.Delete(path);
                Request.Files["file-
upload"].SaveAs(path);
            }
            db.Replays.Add(new Replay { ReplayID =
ticks.ToString(), Title =
Request["Title"].ToString(), IsPublic = false,
UserName = User.Identity.GetUserName(), LastModified
= DateTime.Now });
            db.SaveChanges();
            return RedirectToAction("index",
"speedoptions", new { refNum = ticks.ToString() });
        }
```

Kode Sumber 4.6 Kelas UploadController

4.2.3. Implementasi Lapisan Antarmuka

Lapisan antarmuka merupakan lapisan yang berisi kelas-kelas yang bertugas untuk menangani tampilan dari setiap halaman yang

dibuka oleh pengguna. Dalam aplikasi CodeReplayer ini, terdapat setidaknya 6 antarmuka yang ditampilkan pada Tabel 4.6.

Tabel 4.6 Daftar Kelas Antarmuka

No	Nama Antarmuka
1.	Embed
2.	Home
3.	Replay
4.	ReplayList
5.	SpeedOptions
6.	Upload

4.2.3.1. Embed

Antarmuka Embed bertugas menangani tampilan untuk halaman fitur *Widget* CodeReplayer. Antarmuka halaman *Widget* pada *web* CodeReplayer ditampilkan pada Gambar 4.1.

Gambar 4.1 Antarmuka fitur Embed CodeReplayer

Pada halaman ini, pengguna dapat melihat *preview* dari hasil *widget* CodeReplayer serta kode yang dapat digunakan jika ingin memasang hasil reka ulang pada halaman *web* pengguna.

4.2.3.2. Home

Antarmuka Home bertugas untuk menangani beberapa tampilan untuk halaman utama dan kontak. Antarmuka halaman tersebut berturut-turut ditampilkan pada Gambar 4.2 dan Gambar 4.3.

Gambar 4.2 Halaman Utama

Gambar 4.3 Halaman Kontak

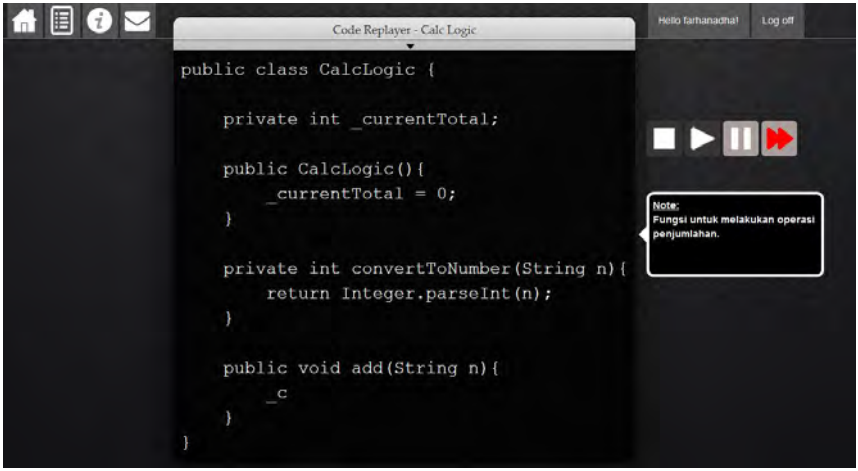
4.2.3.3. Replay

Antarmuka Replay bertugas untuk menangani tampilan ketika pengguna melakukan reka ulang. Pada antarmuka Replay, terdapat pula kode Javascript yang digunakan untuk melakukan reka ulang seperti yang ditunjukkan pada Lampiran A. Sedangkan antarmuka dari reka ulang ditampilkan pada Gambar 4.4 dan Gambar 4.5.

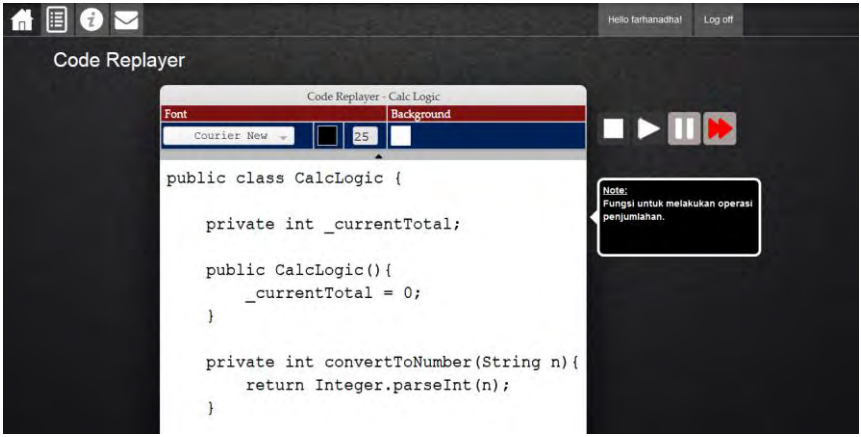
4.2.3.4. ReplayList

Antarmuka ReplayList bertugas untuk menangani tampilan pada halaman kelola berkas XML. Pada antarmuka ReplayList

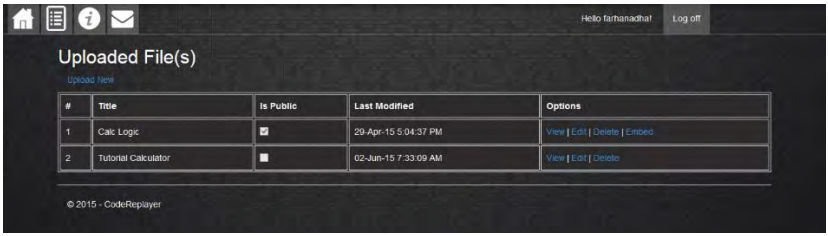
terdapat beberapa menu seperti unggah, sunting, hapus, *embed*, dan lihat reka ulang. Antarmuka ReplayList ditampilkan pada Gambar 4.6.



Gambar 4.4 Antarmuka Reka Ulang (1)

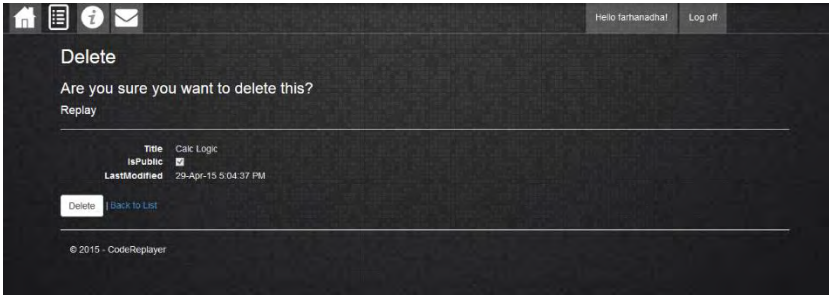


Gambar 4.5 Antarmuka Reka Ulang (2)



Gambar 4.6 Antarmuka Kelola Berkas XML

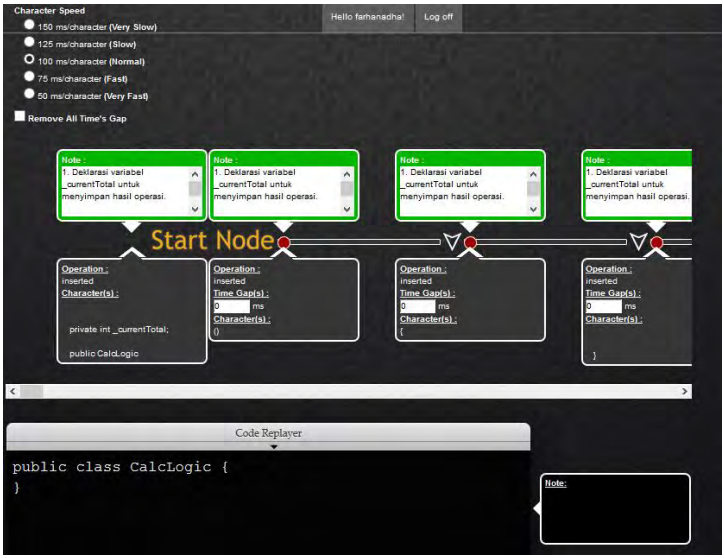
Jika pengguna ingin menghapus salah satu berkas XML yang telah diunggah, pengguna dapat menekan tombol hapus pada berkas XML yang bersangkutan dan akan diarahkan ke halaman konfirmasi sebelum menghapus, seperti tampilan pada Gambar 4.7.



Gambar 4.7 Antarmuka Hapus Berkas XML

4.2.3.5. SpeedOptions

Antarmuka SpeedOptions menangani tampilan pada saat pengguna melakukan operasi penyuntingan berkas XML. Pada operasi penyuntingan, beberapa pilihan yang dapat dipilih oleh pengguna seperti: kecepatan karakter, penambahan catatan, dan menghapus jeda antar node. Selain itu, ditampilkan pula *preview* reka ulang dari hasil sunting yang dilakukan pengguna Untuk detilnya, antarmuka SpeedOptions ditampilkan pada Gambar 4.8.



Gambar 4.8 Antarmuka Sunting Berkas XML (1)

Jika pengguna sudah selesai melakukan penyuntingan dan ingin melihat hasil reka ulang, pengguna dapat menekan tombol Save yang berada di pojok kanan bawah seperti pada Gambar 4.9.

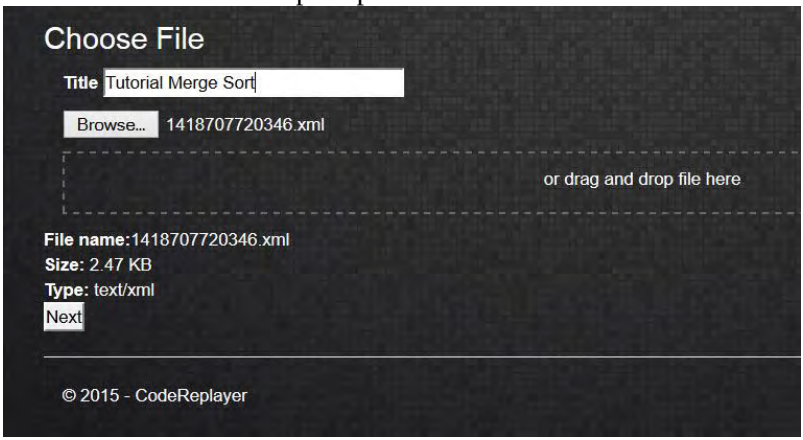
Gambar 4.9 Antarmuka Sunting Berkas XML (2)

4.2.3.6. Upload

Antarmuka Upload digunakan untuk menangani fitur mengunggah berkas XML. Pada antarmuka Upload ditampilkan *form* yang harus dilengkapi ketika pengguna mengunggah berkas XML seperti ditampilkan pada Gambar 4.10.

Gambar 4.10 Antarmuka Unggah Berkas XML (1)

Jika pengguna sudah melengkapi *form*, pengguna dapat menekan tombol Next seperti pada Gambar 4.11.



Gambar 4.11 Antarmuka Unggah Berkas XML (2)

BAB V
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tentang pengujian dan evaluasi dari aplikasi yang dikembangkan. Pengujian dilakukan pada tiap fitur dan kode program CodeReplayer. Hasil evaluasi menjelaskan mengenai rangkuman pengujian yang dilakukan pada aplikasi ini.

5.1. Lingkungan Pengujian

Lingkungan pengujian tugas akhir ini adalah sebagai berikut:

Prosesor	: Intel Core i5 4200M CPU @2.50 GHz 2.50Ghz
Memori	: 8.00 GB
Jenis Device	: Komputer Lenovo
Sistem Operasi	: Microsoft Windows 8.1 64-bit
Tools	: Microsoft Visual Studio 2013 Eclipse Java EE 4.4.1
Framework	: ASP.NET MVC 5
	: Entity *Framework* 6
Server	: IIS 8 Express

5.2. Pengujian *black box*

Pada subbab ini dijelaskan pengujian aplikasi CodeReplayer dengan metode *black box*. Metode ini cenderung melihat pada hasil keluaran aplikasi dari tiap fungsionalitas atau kasus penggunaan yang ada pada aplikasi CodeReplayer. Tiap kasus penggunaan akan diuji dengan beberapa skenario kasus uji.

5.2.1. Pengujian Fitur Mengunggah Berkas XML

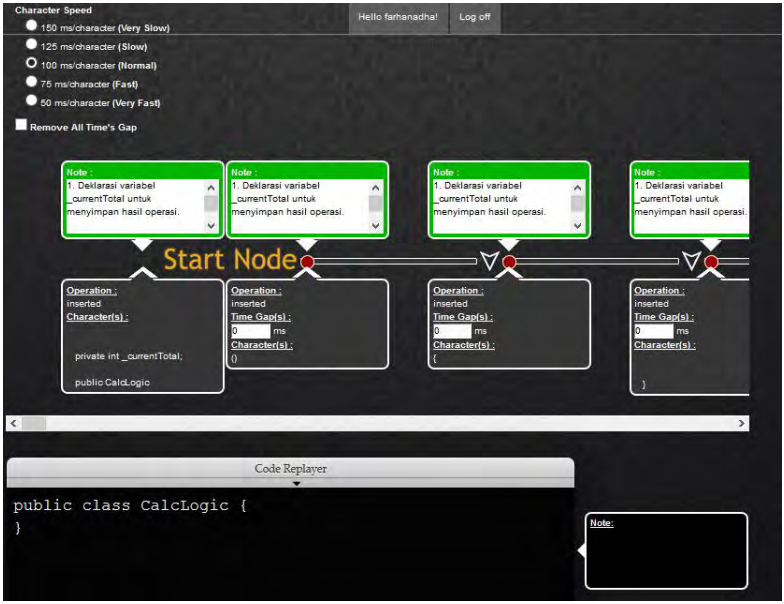
Pengujian fitur mengunggah berkas XML merupakan pengujian terhadap kemampuan sistem menangani fasilitas unggah berkas XML yang merupakan output dari plug-in OperationRecorder pada IDE Eclipse.

5.2.1.1. Skenario 1 Pengujian Fitur Mengunggah Berkas XML

Pada skenario ini, fitur mengunggah berkas XML akan diuji dengan *input* berupa berkas XML hasil dari *output plug-in* OperationRecorder. Detil dari skenario ini akan dijelaskan dalam Tabel 5.1.

Tabel 5.1 Skenario 1 Uji Fitur Mengunggah Berkas XML

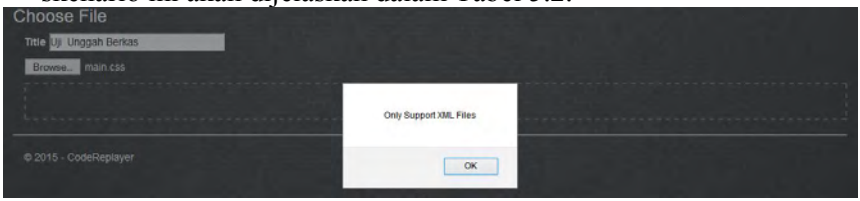
ID	UC-01-TS-01
Referensi Kasus Penggunaan	UC-01
Nama	Pengujian fitur mengunggah berkas XML
Tujuan Pengujian	Menguji fitur untuk mengunggah berkas XML hasil *plug-in* OperationRecorder.
Skenario	**Pengguna mengunggah berkas XML hasil plug-in OperationRecorder.**
Kondisi Awal	Pengguna memiliki berkas XML hasil plug-in OperationRecorder.
Data Uji	**Data uji berupa berkas XML hasil dari plug-in OperationRecorder.**
Langkah Pengujian	1. Pengguna masuk ke halaman *Home* CodeReplayer. 2. Pengguna mengisi judul dari berkas XML yang diunggah. 3. Pengguna memilih berkas XML yang ingin diunggah dengan meng-klik tombol "*Browse*". 4. Pengguna menekan tombol "Next".
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman sunting berkas XML dan menampilkan data dari berkas XML tersebut.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman sunting berkas XML dan menampilkan data dari berkas XML tersebut.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna diarahkan ke halaman sunting berkas XML, seperti pada Gambar 5.1.



Gambar 5.1 Hasil Uji Mengunggah Berkas XML Skenario 1

5.2.1.2. Skenario 2 Pengujian Fitur Mengunggah Berkas XML

Pada skenario ini, fitur mengunggah berkas XML akan diuji dengan *input* yang bukan merupakan berkas XML. Detil dari skenario ini akan dijelaskan dalam Tabel 5.2.



Gambar 5.2 Hasil Uji Mengunggah Berkas XML Skenario 2

Tabel 5.2 Skenario 2 Uji Fitur Mengunggah Berkas XML

ID	UC-01-TS-02
Referensi Kasus Penggunaan	UC-01
Nama	Pengujian fitur mengunggah berkas XML
Tujuan Pengujian	Menguji fitur untuk mengunggah berkas XML dengan input yang tidak valid (bukan berkas XML).
Skenario	**Pengguna mengunggah berkas selain berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan memiliki berkas selain berkas XML.
Data Uji	**Data uji berupa berkas lain selain berkas XML.**
Langkah Pengujian	1. Pengguna masuk ke halaman *Home* CodeReplayer. 2. Pengguna mengisi judul pada kolom isian judul. 3. Pengguna memilih berkas selain berkas XML.
Hasil Yang Diharapkan	Sistem menampilkan notifikasi bahwa hanya berkas XML yang diperbolehkan untuk diunggah.
Hasil Yang Didapat	Sistem menampilkan notifikasi bahwa hanya berkas XML yang diperbolehkan untuk diunggah.
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem memberikan notifikasi bahwa hanya berkas XML yang diperbolehkan untuk diunggah,seperti pada Gambar 5.2.

5.2.2. Pengujian Fitur Menyunting Hasil Perekaman

Pengujian fitur menyunting hasil perekaman merupakan pengujian terhadap kemampuan sistem menangani fasilitas sunting berkas XML setelah berkas XML tersebut diunggah. Fasilitas sunting berkas XML ini meliputi: pemilihan kecepatan tiap karakter, penyuntingan jeda waktu antar *node* pada berkas XML, penambahan catatan atau dokumentasi pada sebuah *node*.

5.2.2.1. Skenario 1 Pengujian Fitur Menyunting Hasil Perekaman

Pada skenario ini, fitur menyunting hasil perekaman akan diuji dengan mengubah atau memilih kecepatan karakter pada halaman sunting berkas XML. Detil dari skenario ini akan dijelaskan dalam Tabel 5.3.

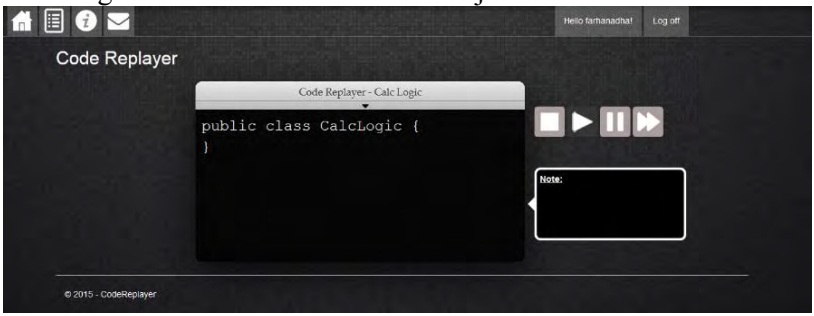
Tabel 5.3 Skenario 1 Uji Fitur Menyunting Hasil Perekaman

ID	UC-02-TS-01
Referensi Kasus Penggunaan	UC-02
Nama	Pengujian fitur menyunting hasil perekaman
Tujuan Pengujian	Menguji fitur untuk menyunting berkas XML dengan mengubah atau memilih kecepatan karakter.
Skenario	**Pengguna memilih salah satu pilihan kecepatan karakter pada halaman sunting berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang sudah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih menu "Edit" pada salah satu berkas XML yang ingin disunting.
Hasil Yang Diharapkan	Sistem menyimpan hasil sunting berkas XML dan mengarahkan pengguna ke halaman lihat reka ulang.
Hasil Yang Didapat	Sistem menyimpan hasil sunting berkas XML dan mengarahkan pengguna ke halaman lihat reka ulang.
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menyimpan hasil sunting berkas XML yang dilakukan pengguna dan mengarahkan pengguna ke halaman lihat reka ulang, seperti pada Gambar 5.3.

Gambar 5.3 Hasil Uji Menyunting Hasil Perekaman Skenario 1

5.2.2.2. Skenario 2 Pengujian Fitur Menyunting Hasil Perekaman

Pada skenario ini, fitur menyunting hasil perekaman akan diuji dengan mengubah atau memilih kecepatan karakter serta mengubah jeda waktu antar *node* pada berkas XML yang disunting. Detil dari skenario ini akan dijelaskan dalam Tabel 5.4.



Gambar 5.4 Hasil Uji Menyunting Hasil Perekaman Skenario 2

Tabel 5.4 Skenario 2 Uji Fitur Meyunting Hasil Perekaman

ID	UC-02-TS-02
Referensi Kasus Penggunaan	UC-02
Nama	Pengujian fitur menyunting hasil perekaman
Tujuan Pengujian	Menguji fitur untuk menyunting berkas XML dengan mengubah atau memilih kecepatan karakter dan mengubah waktu jeda antar node.
Skenario	**Pengguna memilih salah satu pilihan kecepatan karakter dan mengubah jeda waktu antar node pada halaman sunting berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang sudah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih menu "Edit" pada salah satu berkas XML yang ingin disunting.
Hasil Yang Diharapkan	Sistem menyimpan hasil sunting berkas XML dan mengarahkan pengguna ke halaman lihat reka ulang.
Hasil Yang Didapat	Sistem menyimpan hasil sunting berkas XML dan mengarahkan pengguna ke halaman lihat reka ulang.
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem menyimpan hasil sunting berkas XML yang dilakukan pengguna dan mengarahkan pengguna ke halaman lihat reka ulang seperti pada Gambar 5.4.

5.2.3. Pengujian Fitur Melihat Reka Ulang

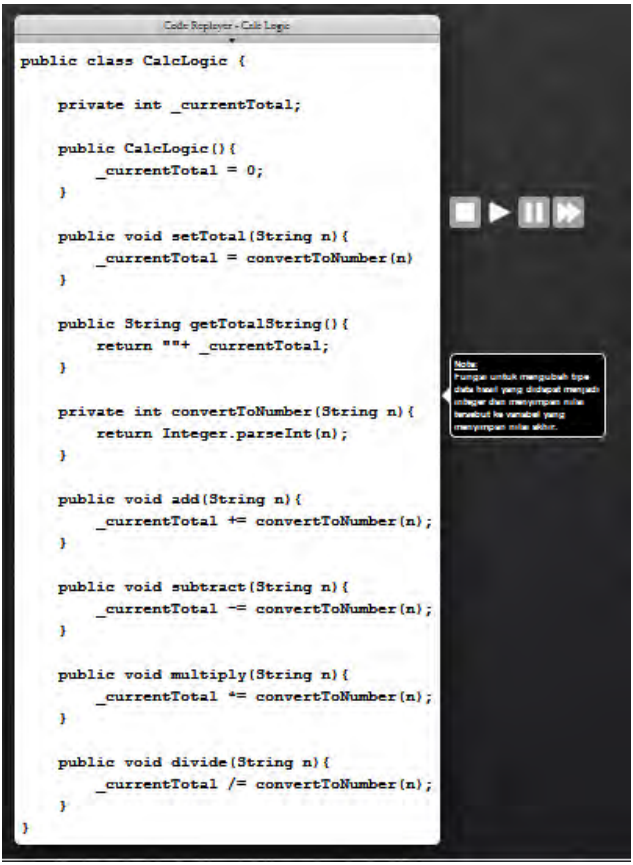
Pengujian fitur melihat reka ulang merupakan pengujian terhadap kemampuan sistem menangani fasilitas reka ulang setelah berkas XML tersebut diunggah dan/atau disunting. Dalam fitur reka ulang, pengguna dapat melakukan kontrol terhadap jalannya proses reka ulang ini, seperti: *stop*, *play*, *pause*, dan *fast forward*.

5.2.3.1. Skenario 1 Pengujian Fitur Melihat Reka Ulang

Pada skenario ini, fitur melihat reka ulang akan diuji dengan memainkan reka ulang dengan memulainya menggunakan tombol *play* dan menguji hasil reka ulang hingga selesai. Detil dari skenario ini akan dijelaskan dalam Tabel 5.5.

Tabel 5.5 Skenario 1 Uji Fitur Melihat Reka Ulang

ID	UC-03-TS-01
Referensi Kasus Penggunaan	UC-03
Nama	Pengujian fitur melihat reka ulang
Tujuan Pengujian	Menguji fitur untuk melihat reka ulang yang dimulai dengan menekan tombol *play* dan melihat hasil reka ulang tersebut hinggal selesai.
Skenario	**Pengguna menekan tombol *play* pada halaman reka ulang untuk memulai dan menunggu proses reka ulang hingga selesai.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang sudah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih menu "View" pada salah satu berkas XML yang ingin disunting.
Hasil Yang Diharapkan	Sistem memainkan hasil reka ulang tersebut sesuai dengan proses pembuatan kode program dan hasil penyuntingan berkas XML runut secara kronologis.
Hasil Yang Didapat	Sistem memainkan hasil reka ulang tersebut sesuai dengan proses pembuatan kode program dan hasil penyuntingan berkas XML runut secara kronologis.
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem selesai memainkan hasil reka ulang, seperti pada Gambar 5.5.



Gambar 5.5 Hasil Uji Melihat Reka Ulang Skenario 1

5.2.3.2. Skenario 2 Pengujian Melihat Reka Ulang

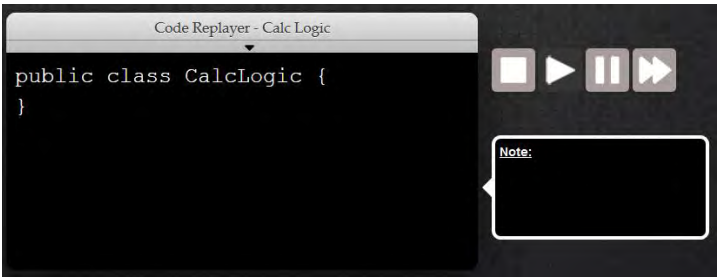
Pada skenario ini, fitur melihat reka ulang akan diuji dengan memainkan reka ulang dengan memulainya menggunakan tombol *play* dan menguji tombol-tombol kontrol lain seperti:

1. Tombol *stop* : Menghentikan reka ulang dan memulai kembali dari awal reka ulang.
2. Tombol *pause* : Menghentikan reka ulang sejenak dan dapat dilanjutkan kembali dengan menekan tombol *play*.

3. Tombol *fast forward* : Mempercepat jalannya reka ulang dua kali lipat dari kecepatan sebelumnya.
   Detil dari skenario ini akan dijelaskan dalam Tabel 5.6.

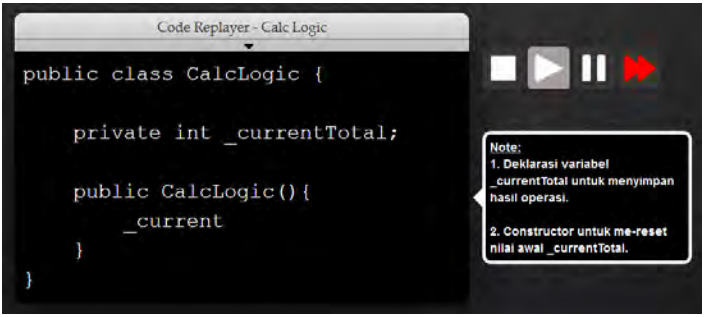
Tabel 5.6 Skenario 2 Uji Fitur Melihat Reka Ulang

ID	UC-03-TS-02
Referensi Kasus Penggunaan	UC-03
Nama	Pengujian fitur melihat reka ulang
Tujuan Pengujian	Menguji fitur untuk melihat reka ulang yang dimulai dengan menekan tombol *play* dan menguji beberapa tombol kontrol seperti *stop*, *pause*, dan *fast forward*.
Skenario	**Pengguna menekan tombol *play* pada halaman reka ulang untuk memulai dan menggunakan tombol kontrol lain seperti *stop*, *pause*, dan *fast forward*.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang sudah diunggah ke sistem.**
Langkah Pengujian	4. Pengguna masuk ke halaman kelola berkas XML. 5. Pengguna memilih menu "View" pada salah satu berkas XML yang ingin disunting.
Hasil Yang Diharapkan	Sistem menampilkan dengan tepat hasil reka ulang dengan kontrol-kontrol seperti *stop*, *pause*, dan *fast forward*.
Hasil Yang Didapat	Sistem menampilkan dengan tepat hasil reka ulang dengan kontrol-kontrol seperti *stop*, *pause*, dan *fast forward*.
Hasil Pengujian	Berhasil
Kondisi Akhir	Sistem memainkan hasil reka ulang, berhenti, atau dipercepat sesuai dengan tombol yang ditekan oleh pengguna, seperti pada Gambar 5.6, Gambar 5.7, dan Gambar 5.8.



Gambar 5.6 Hasil Uji Melihat Reka Ulang Skenario 2

Gambar 5.7 Hasil Uji Melihat Reka Ulang Skenario 2(1)



Gambar 5.8 Hasil Uji Melihat Reka Ulang Skenario 2(2)

5.2.4. Pengujian Fitur Mengelola Berkas XML

Pengujian fitur mengelola berkas XML merupakan pengujian terhadap kemampuan sistem menangani fasilitas pengelolaan berkas-berkas XML setelah berkas tersebut diunggah. Dalam fitur

mengelola berkas XML, pengguna dapat melihat reka ulang, menghapus, menyunting, dan melakukan *embed* hasil reka ulang berkas XML serta menambah/mengunggah berkas XML baru.

5.2.4.1. Skenario 1 Pengujian Fitur Mengelola Berkas XML

Pada skenario ini, fitur mengelola berkas akan diuji dengan menekan tombol untuk mengunggah berkas XML baru dari halaman kelola berkas. Detil dari skenario ini akan dijelaskan dalam Tabel 5.7.

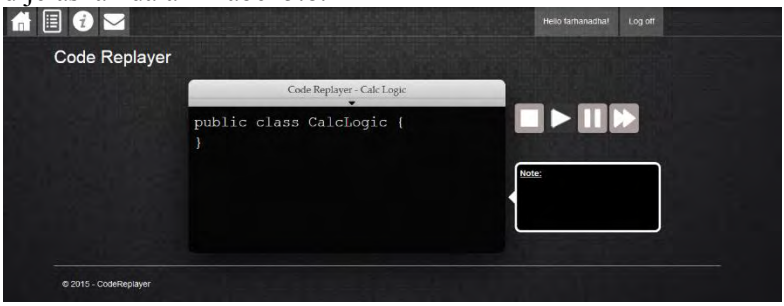
Tabel 5.7 Skenario 1 Uji Fitur Mengelola Berkas XML

ID	UC-04-TS-01
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian fitur mengelola berkas XML
Tujuan Pengujian	Menguji fitur untuk menambah/mengunggah berkas XML baru dari halaman kelola berkas XML.
Skenario	**Pengguna menekan tombol *"Upload New"* pada halaman kelola berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan berada pada halaman kelola berkas XML.
Data Uji	**Data uji berupa *request* pengguna ke halaman unggah berkas XML.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih tombol "*Upload New*" pada halaman kelola berkas XML.
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman unggah berkas XML.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman unggah berkas XML.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna masuk ke halaman unggah berkas XML, seperti pada Gambar 5.9.

Gambar 5.9 Hasil Uji Mengelola Berkas XML Skenario 1

5.2.4.2. Skenario 2 Pengujian Fitur Mengelola Berkas XML

Pada skenario ini, fitur mengelola berkas akan diuji dengan menekan tombol untuk melihat reka ulang pada berkas XML yang dipilih dari halaman kelola berkas. Detil dari skenario ini akan dijelaskan dalam Tabel 5.8.



Gambar 5.10 Hasil Uji Mengelola Berkas XML Skenario 2

Tabel 5.8 Skenario 2 Uji Fitur Mengelola Berkas XML

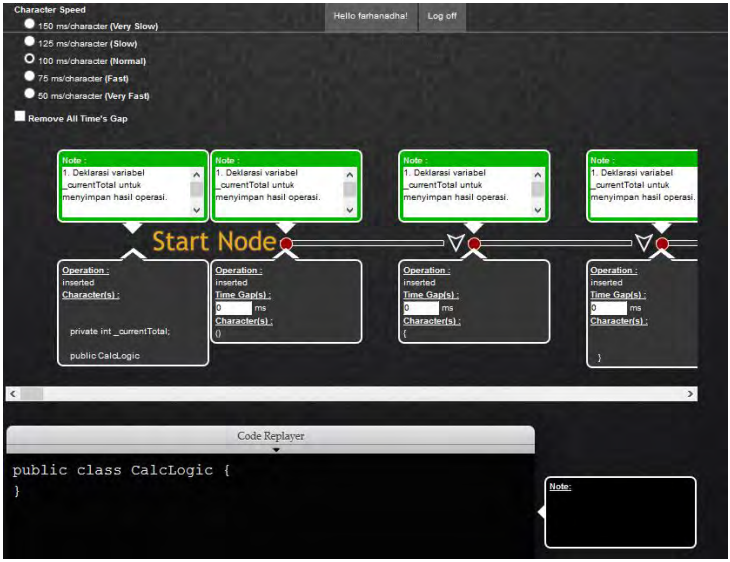
ID	UC-04-TS-02
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian fitur mengelola berkas XML
Tujuan Pengujian	Menguji fitur untuk melihat hasil reka ulang berkas XML melalui halaman kelola berkas XML.
Skenario	**Pengguna menekan tombol *"View"* pada berkas XML yang ingin dilihat hasil reka ulangnya di halaman kelola berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan sudah mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang telah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih tombol "*View*" pada berkas XML yang ingin dilihat hasil reka ulangnya.
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman melihat reka ulang.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman melihat reka ulang.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna masuk ke halaman melihat reka ulang, seperti pada Gambar 5.10.

5.2.4.3. Skenario 3 Pengujian Fitur Mengelola Berkas XML

Pada skenario ini, fitur mengelola berkas akan diuji dengan menekan tombol untuk menyunting berkas XML yang dipilih dari halaman kelola berkas. Detil dari skenario ini akan dijelaskan dalam Tabel 5.9.

Tabel 5.9 Skenario 3 Uji Fitur Mengelola Berkas XML

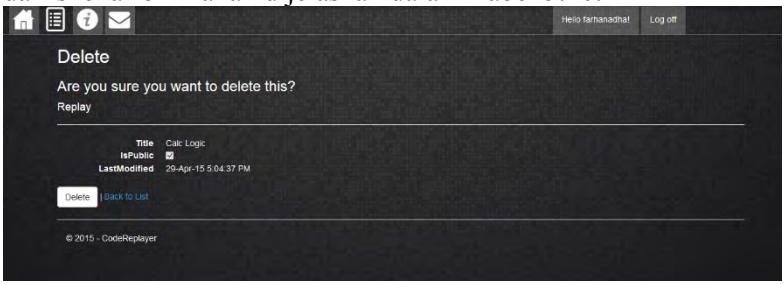
ID	UC-04-TS-03
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian fitur mengelola berkas XML
Tujuan Pengujian	Menguji fitur untuk menyunting kembali berkas XML yang sebelumnya telah diunggah.
Skenario	**Pengguna menekan tombol *"Edit"* pada berkas XML yang ingin disunting kembali di halaman kelola berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan sudah mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang telah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih tombol "*Edit*" pada berkas XML yang ingin disunting.
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman sunting berkas XML.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman sunting berkas XML.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna masuk ke halaman sunting berkas XML sesuai dengan hasil sunting yang telah tersimpan, seperti pada Gambar 5.11.



Gambar 5.11 Hasil Uji Mengelola Berkas XML Skenario 3

5.2.4.4. Skenario 4 Pengujian Fitur Mengelola Berkas XML

Pada skenario ini, fitur mengelola berkas akan diuji dengan menekan tombol untuk menghapus berkas XML yang dipilih. Detil dari skenario ini akan dijelaskan dalam Tabel 5.10.



Gambar 5.12 Hasil Uji Mengelola Berkas XML Skenario 4

Tabel 5.10 Skenario 4 Uji Fitur Mengelola Berkas XML

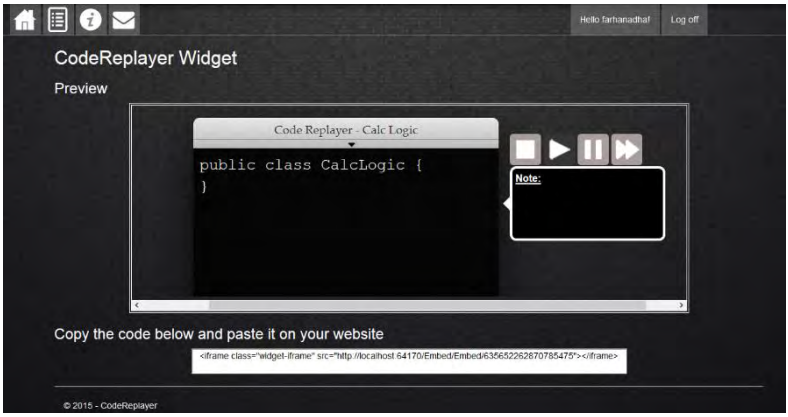
ID	UC-04-TS-04
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian fitur mengelola berkas XML
Tujuan Pengujian	Menguji fitur untuk menghapus berkas XML yang sebelumnya telah diunggah.
Skenario	**Pengguna menekan tombol *"Delete"* pada berkas XML yang ingin dihapus di halaman kelola berkas XML.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan sudah mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang telah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih tombol "*Delete*" pada berkas XML yang ingin dihapus.
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman konfirmasi untuk menghapus berkas XML.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman konfirmasi untuk menghapus berkas XML.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna dikonfirmasi sebelum berkas XML dihapus, seperti pada Gambar 5.12. Dan pengguna dikembalikan ke halaman kelola berkas XML dengan tabel berbeda yang telah menghapus berkas XML yang dipilih.

5.2.4.5. Skenario 5 Pengujian Fitur Mengelola Berkas XML

Pada skenario ini, fitur mengelola berkas akan diuji dengan menekan tombol untuk memasang fitur widget atau melakukan embed dari hasil reka ulang pada berkas XML yang dipilih. Detil dari skenario ini akan dijelaskan dalam Tabel 5.11.

Tabel 5.11 Skenario 5 Uji Fitur Mengelola Berkas XML

ID	UC-04-TS-05
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian fitur mengelola berkas XML
Tujuan Pengujian	Menguji fitur untuk memasang widget hasil reka ulang berkas XML yang sebelumnya telah diunggah.
Skenario	**Pengguna menekan tombol *"Embed"* pada berkas XML yang ingin di-*embed*.**
Kondisi Awal	Pengguna sudah login kedalam sistem dan sudah mengunggah berkas XML.
Data Uji	**Data uji berupa berkas XML yang telah diunggah ke sistem.**
Langkah Pengujian	1. Pengguna masuk ke halaman kelola berkas XML. 2. Pengguna memilih tombol "*Embed*" pada berkas XML yang ingin di-*embed*.
Hasil Yang Diharapkan	Sistem mengarahkan pengguna ke halaman Embed untuk memasang widget hasil reka ulang berkas XML.
Hasil Yang Didapat	Sistem mengarahkan pengguna ke halaman Embed untuk memasang widget hasil reka ulang berkas XML.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna diarahkan ke halaman *Embed* hasil reka ulang, seperti pada Gambar 5.13.



Gambar 5.13 Hasil Uji Mengelola Berkas XML Skenario 5

5.2.5. Hasil Pengujian *black box*

Berdasarkan hasil pengujian dengan metode *black box*, didapat bahwa semua fungsional sistem mampu berjalan dengan baik, menerima *input* dan menghasilkan *output* sesuai dengan hasil yang diharapkan.

5.3. Pengujian *white box*

Pada subbab ini dijelaskan pengujian aplikasi CodeReplayer dengan metode *white Box*. Metode ini cenderung melihat pada kode program untuk menganalisis kualitas kode program. *white box* testing dilakukan dengan beberapa kakas bantu seperti CodeMetrics dan NDepend.

5.3.1. Pengujian dengan CodeMetrics

CodeMetrics melakukan pengujian terhadap kode sumber dengan beberapa parameter berikut [12]:

1. ***Maintainability Index*** : Nilai yang merepresentasikan derajat kemudahan perawatan, semakin tinggi nilai semakin mudah melakukan perawatan. Nilai berkisar antara 1-100.
2. ***Cyclomatic Complexity*** : Merupakan bilangan yang merepresentasikan tingkat kompleksitas kode program yang

dihitung dengan jumlah *path* berbeda yang mungkin dilalui oleh program.

3. ***Depth of Inheritance*** : Mengindikasikan jumlah terbanyak sebuah kelas diturunkan dalam sebuah kelas.
4. ***Class Coupling*** : Merupakan derajat ketergantungan sebuah kelas dengan kelas lainnya. Perangkat lunak yang baik memiliki nilai Class Coupling yang rendah.

Hasil dari pengujian dengan CodeMetrics ditunjukkan pada Tabel 5.12.

Tabel 5.12 Hasil Uji CodeMetrics

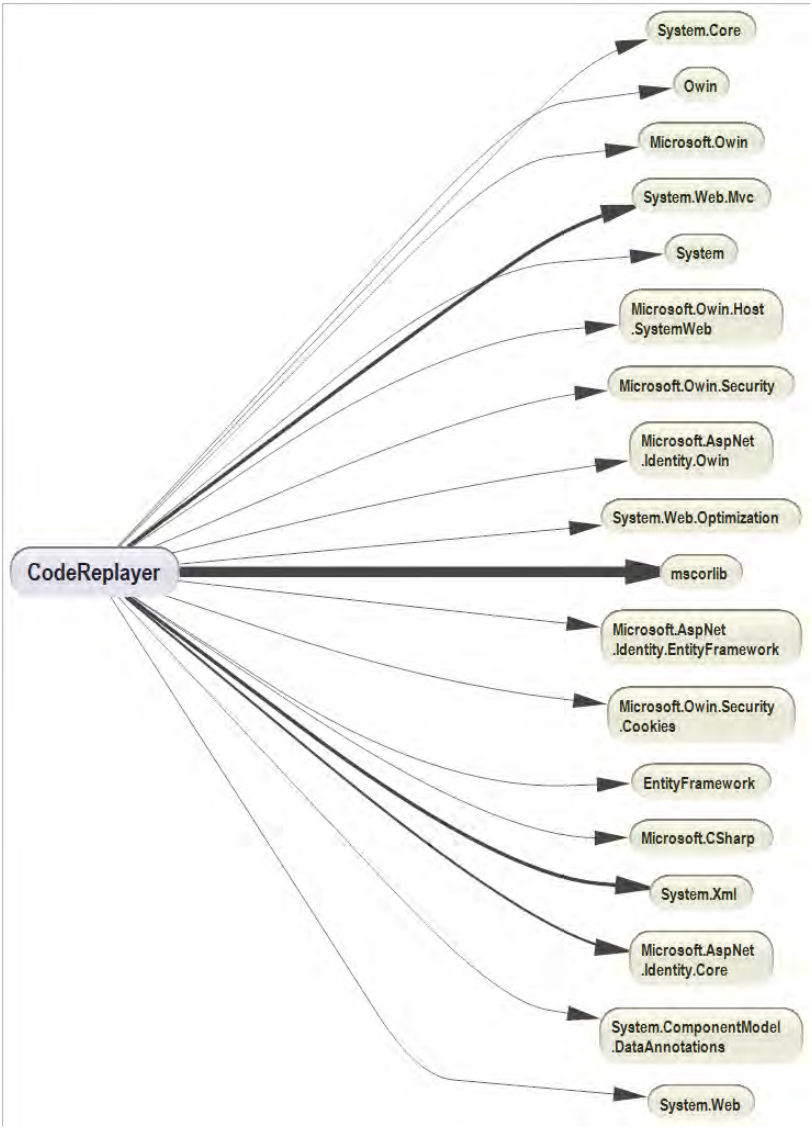
Hierarchy	*Maintainability Index*	*Cyclomatic Complexity*	*Depth of Inheritance*	*Class Coupling*
CodeReplayer Solution	87	274	4	139
CodeReplayer	86	11	2	28
CodeReplayer Controllers	77	173	4	101
CodeReplayer Models	94	90	3	20

5.3.2. Pengujian dengan NDepend

Pada pengujian dengan kakas bantu NDepend, dapat diketahui *program dependency graph* dan *program dependency matrix* dari aplikasi CodeReplayer.

5.3.2.1. *Program Dependency Graph*

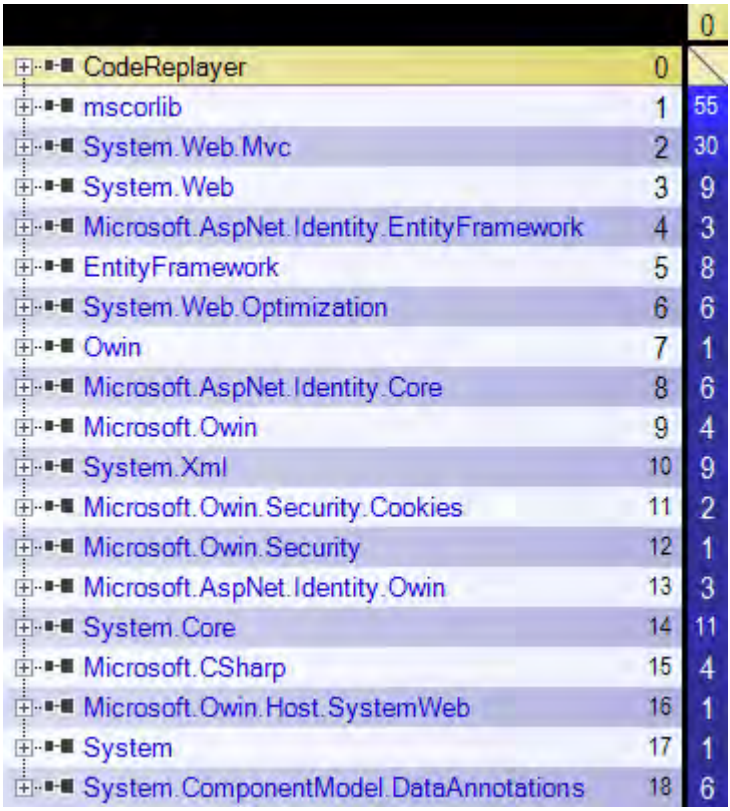
Program Dependency Graph, yang direpresentasikan pada Gambar 5.14, mengindikasikan modul-modul (*assembly*/dll) dari luar yang digunakan di dalam program. Pada Gambar 5.14. menunjukkan bahwa semakin tebal panah maka semakin banyak pula referensi pada modul tersebut digunakan dalam kode program CodeReplayer.



Gambar 5.14 ***Program Dependency Graph***

5.3.2.2. *Program Dependency Matrix*

Program Dependency Matrix merupakan cara merepresentasikan Program Dependency Graph dengan matrix. Seperti halnya Program Dependency Graph, Program Dependency Matrix juga mengindikasikan modul-modul (*assembly*/dll) dari luar yang digunakan di dalam program. Pada Gambar 5.15, menunjukkan pada kolom terakhir dari matrix tersebut mengindikasikan seberapa banyak referensi dari sebuah modul digunakan pada program CodeReplayer.



Gambar 5.15 ***Program Dependency Matrix***

5.3.3. Hasil Pengujian *white box*

Berdasarkan hasil pengujian dengan metode *white box*, dengan pengujian CodeMetrics didapat bahwa nilai *Maintainability Index* sebesar 87/100, *Cyclomatix Complexity* sebesar 274, *Depth of Inheritance* maksimal adalah 4, dan *Class Coupling* sebesar 139. Sedangkan, dari hasil pengujian dengan NDepend, didapat bahwa CodeReplayer memiliki ketergantungan terhadap setidaknya 18 modul lain, dengan ketergantungan terbanyak pada modul mscorlib (sebanyak 55 referensi) yang berisi librari dari kelas-kelas dasar C#.

BAB VI
KESIMPULAN DAN SARAN

Bab berikut menjelaskan terkait kesimpulan dan saran -saran yang didapatkan dari hasil proses pembuatan tugas akhir.

6.1. Kesimpulan

Dalam proses pengerjaan dari tahap analisis, perancangan, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Aplikasi CodeReplayer dapat membaca dan mengolah berkas XML yang merupakan keluaran dari *plug-in* OperationRecorder pada IDE Eclipse.
2. Aplikasi CodeReplayer dapat melakukan reka ulang secara tepat dan kronologis berdasarkan proses pembuatan kode program yang sebelumnya direkam oleh *plug-in* OperationRecorder.
3. Aplikasi CodeReplayer dapat melakukan penyuntingan pada berkas XML OperationRecorder seperti mengubah waktu jeda antar *node* dan menambahkan fitur dokumentasi/catatan untuk tiap *node*.

6.2. Saran

Adapun saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang, diantaranya adalah sebagai berikut:

1. Adanya *syntax higlighting* pada saat melakukan reka ulang untuk mempermudah mengenali bagian-bagian pada kode program.
2. Pembuatan *plug-in* yang melakukan perekaman agar antara *plug-in* dan aplikasi CodeReplayer lebih terintegrasi sehingga pengembangan selanjutnya lebih mudah dilakukan.

DAFTAR PUSTAKA

[1] T. Omori dan K. Maruyama, "A Change-Aware Development Environment by Recording Editing Operations of Source Code," *Proceedings of the 2008 international working conference on Mining software repositories*, pp. 31-34, 2008.

[2] T. Omori dan K. Maruyama, "An Editing-operation Replayer with Highlights Supporting," *IWPSE-EVOL '11*, 2011.

[3] Microsoft, "ASP.Net MVC," [Online]. Available: http://www.asp.net/mvc. [Diakses 2014].

[4] Microsoft, "ASP.NET," [Online]. Available: http://www.asp.net/. [Diakses 2014].

[5] Bootstrap, "Bootstrap," [Online]. Available: http://getbootstrap.com/. [Diakses 2014].

[6] Microsoft, "ASP.NET Data Access Options," [Online]. Available: https://msdn.microsoft.com/en-us/library/ms178359%28v=vs.110%29.aspx. [Diakses 2014].

[7] Microsoft, "XMLDocument Class," 2014. [Online]. Available: https://msdn.microsoft.com/en-us/library/system.xml.xmldocument%28v=vs.100%29.aspx.

[8] Microsoft, "XMLSerializer Class," 2014. [Online]. Available: https://msdn.microsoft.com/en-us/library/system.xml.xmldocument%28v=vs.100%29.aspx.

[9] Javascript, "Javascript," [Online]. Available: https://www.javascript.com/. [Diakses 2014].

[10] jQuery, "jQuery," [Online]. Available: https://jquery.com/. [Diakses 2014].

[11] JSON, "Introducing JSON," [Online]. Available: http://json.org/. [Diakses 2014].

[12] Microsoft, "Code Metrics Values," [Online]. Available: https://msdn.microsoft.com/en-us/library/bb385914.aspx. [Diakses 2015].

[13] T. Omori, "OperationRecorder," [Online]. Available: http://www.ritsumei.ac.jp/~tomori/operec.html. [Diakses 2014].

[14] IBM, "UML Basics : The Sequence Diagram," [Online]. Available: http://www.ibm.com/developerworks/rational/library/3101.html. [Diakses 2015].

[15] Microsoft, "ASP.NET Entity Framework," 2014. [Online]. Available: http://www.asp.net/entity-framework.

LAMPIRAN A. KODE SUMBER

```
<script>
    function showHideProp() {
        if
(document.getElementById('prop').style.visibility ==
'collapse') {
            $('#prop').css('visibility', 'visible');

document.getElementById('buttonShow').style.backgroun
dImage = "url(/Picture/arrowup.png)";

        }
        else {
            $('#prop').css('visibility', 'collapse');

document.getElementById('buttonShow').style.backgroun
dImage = "url(/Picture/arrowdown.png)";
        }
    }

    $("#number").selectmenu({
        change: function () {
            var selectedIndex = $("#number").val();

document.getElementById("divauto").style.fontSize =
selectedIndex + "px";
        }
    });

    $('#fontSelect').fontSelector({
        'hide_fallbacks': true,
        'initial': 'Courier New,Courier
New,Courier,monospace',
        'selected': function (style) {

document.getElementById("divauto").style.fontFamily =
style;
        },
        'fonts': [
```

```
        'Arial,Arial,Helvetica,sans-serif',
        'Arial Black,Arial Black,Gadget,sans-serif',
        'Comic Sans MS,Comic Sans MS,cursive',
        'Courier New,Courier New,Courier,monospace',
        'Georgia,Georgia,serif',
        'Impact,Charcoal,sans-serif',
        'Lucida Console,Monaco,monospace',
        'Lucida Sans Unicode,Lucida Grande,sans-
serif',
        'Palatino Linotype,Book
Antiqua,Palatino,serif',
        'Tahoma,Geneva,sans-serif',
        'Times New Roman,Times,serif',
        'Trebuchet MS,Helvetica,sans-serif',
        'Verdana,Geneva,sans-serif',
        'Gill Sans,Geneva,sans-serif'
        ]
    });
    $('#font_color_picker').colpick({
        colorScheme: 'dark',
        layout: 'rgbhex',
        color: 'F0F0F0',
        onSubmit: function (hsb, hex, rgb, el) {
            $(el).css('background-color', '#' + hex);

document.getElementById("divauto").style.color = '#'
+ hex;
            $(el).colpickHide();
        }
    })
.css('background-color', '#F0F0F0');

    $('#bg_color_picker').colpick({
        colorScheme: 'dark',
        layout: 'rgbhex',
        color: '000000',
        onSubmit: function (hsb, hex, rgb, el) {
            $(el).css('background-color', '#' + hex);
            document.getElementById("text-
body").style.backgroundColor = '#' + hex;
```

```
            $(el).colpickHide();
        }
    })
.css('background-color', '#000000');

    Element.prototype.documentOffsetTop = function ()
{
        return this.offsetTop + (this.offsetParent ?
this.offsetParent.documentOffsetTop() : 0);
    };

    var stop = 0;
    var pause = 0;
    var pauseCount = 0;
    var to = 0;
    var isFFOn = 0;
    var node;
    var lastNode;
    var charAt2;
    var lastChar;
    var resumeOffset;
    var len;
    var resp;
    var startNode;
    var actionUrl;
    var speedChar;
    var showNote;


    window.onload = doOnLoad;

    function doOnLoad() {
        actionUrl = '@Url.Action("GetJsonOperations",
"SpeedOptions", new { refNum = @Model })';
        $.getJSON(actionUrl, displayData);

document.getElementById('prop').style.visibility =
'collapse';
    }
```

```
    $("#btnPlay").click(function () {
        var pc;
        visibilityIcon(false, true, false, false);

        if (pause == 1 && stop == 0) {
            pause = 0;
            resumeTyping();
        }
        else if (stop > 0) {
            if (stop == 1)
                pause = 0;
            else if (stop == 2) {

document.getElementById("divauto").innerHTML = "";
                displayData(resp);
            }
            stop = 0;
            pc = pauseCount;
        }
        if (pauseCount == 0 || pc == pauseCount) {
            isFFOn = 0;
            startPlay();
        }
    })

    $("#btnPause").click(function () {
        visibilityIcon(false, false, true, true);
        pauseCount++;
        pause = 1;
    })

    $("#btnFastForward").click(function () {
        visibilityIcon(false, true, false, false);
        pauseCount++;
        if (isFFOn == 0) {

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-red.png)";
            speedChar /= 2;
            isFFOn = 1;
```

```
        }
        else {

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
            speedChar *= 2;
            isFFOn = 0;
        }
        resumeTyping();
    })

    $("#btnStop").click(function () {
        pauseCount++;
        stop = 1;

document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
        visibilityIcon(true, false, true, true);
        document.getElementById("divauto").innerHTML
= "";

document.getElementById("noteDialog").innerHTML = "";
        displayData(resp);
    })

    function visibilityIcon(stop, play, pause,
fastForward) {
        document.getElementById("btnStop").disabled =
stop;
        document.getElementById("btnPlay").disabled =
play;
        document.getElementById("btnPause").disabled
= pause;

document.getElementById("btnFastForward").disabled =
fastForward;
    }

    function stopClear() {
        stop = 2;
```

```
document.getElementById("btnFastForward").style.backg
roundImage = "url(/Picture/fastforward-white.png)";
        visibilityIcon(true, false, true, true);
    }

    function createSpan(num) {
        for (var i = 0; i <= num; i++) {
            $('#divauto').append("<span
class='spanChar' id=" + i + "></span>");
        }
    }

    function initStartCode(response) {
        var tempStartNode;
        for (var i = 0; ; i++) {
            if (response.Operations[i].Time != 0) {
                tempStartNode = i;
                break;
            }
            var offs = response.Operations[i].Offset;
            var txt =
response.Operations[i].Character;
            for (var n = 0; n < txt.length; n++) {
                if (txt[n] == '#' && txt[n + 1] ==
'n' && txt[n + 2] == '#') {
                    $('#' + offs).append("<br/>");
                    offs += 2;
                    n += 2;
                }
                else if (txt[n] == '#' && txt[n + 1]
== 't' && txt[n + 2] == '#') {
                    $('#' +
offs).append("   ");
                    offs += 1;
                    n += 2;
                } else if (txt[n] == '#' && txt[n +
1] == 's' && txt[n + 2] == '#') {
                    $('#' + offs).append(" ");
                    offs += 1;
```

```
                    n += 2;
                }
                else {
                    $('#' + offs).append(txt[n]);
                    offs++;
                }
            }
        }
        startNode = tempStartNode;
    }

    function displayData(response) {
        resp = response;
        document.getElementById("titleCR").innerHTML
= "Code Replayer";
        document.getElementById("titleCR").innerHTML
+= " - " + response.Title;
        maxSpan = response.MaxOffset;
        lastNode = resp.Operations.length - 1;
        lastChar =
resp.Operations[lastNode].Character.length - 1;
        createSpan(maxSpan);
        initStartCode(response);
    }

    function startPlay() {
        var pc = pauseCount;
        var nextTo;
        speedChar = resp.Speed;
        to = 300;
        len = resp.Operations.length;

        for (var j = startNode; j < len; j++) {
            appendNoteDialog(j, pc);
            if (resp.Operations[j].OperationName ==
"inserted")
                inserted(j, 0, to,
resp.Operations[j].Character,
resp.Operations[j].Offset, pc);
```

```
            else if (resp.Operations[j].OperationName
== "deleted")
                deleted(j,
resp.Operations[j].Character.length - 1, to,
resp.Operations[j].Character,
resp.Operations[j].Offset - 1, pc);
            if (j < len - 1) {
                nextTo =
(resp.Operations[j].CurrDeltaOffset -
resp.Operations[j].LineNumber) * speedChar;
                to += (resp.Operations[j + 1].Time +
nextTo);
            }
        }
    }


    function resumeTyping() {

        var pc = pauseCount;
        var nextTo;
        to = 0;

        appendNoteDialog(node, pc);
        if (resp.Operations[node].OperationName ==
"inserted")
            inserted(node, charAt2 + 1, to,
resp.Operations[node].Character, resumeOffset, pc);
        else if (resp.Operations[node].OperationName
== "deleted")
            deleted(node, charAt2 - 1, to,
resp.Operations[node].Character, resumeOffset - 1,
pc);
        if (node < len - 1) {
            nextTo =
(resp.Operations[node].CurrDeltaOffset -
resp.Operations[node].LineNumber) * speedChar;
            to += (resp.Operations[node + 1].Time +
nextTo);
        }
```

```
        for (var j = node + 1; j < len; j++) {
            appendNoteDialog(j, pc);
            if (resp.Operations[j].OperationName ==
"inserted")
                inserted(j, 0, to,
resp.Operations[j].Character,
resp.Operations[j].Offset, pc);
            else if (resp.Operations[j].OperationName
== "deleted")
                deleted(j,
resp.Operations[j].Character.length - 1, to,
resp.Operations[j].Character,
resp.Operations[j].Offset - 1, pc);
            if (j < len - 1) {
                nextTo =
(resp.Operations[j].CurrDeltaOffset -
resp.Operations[j].LineNumber) * speedChar;
                to += (resp.Operations[j + 1].Time +
nextTo);
            }
        }
    }

    function appendNoteDialog(num, pc) {
        showNote = setTimeout(function () {
            if (pc == pauseCount)

document.getElementById("noteDialog").innerHTML =
resp.Operations[num].Note;
        }, to);
    }

    function inserted(j, charAt, timeOut, txt,
offset, pc) {
        var delay = speedChar;
        var to = timeOut;
        offset = parseInt(offset);
        for (var i = charAt; i < txt.length; i++) {
```

```
            var temp = doInsert(j, i, txt, to,
offset, pc);
            i = temp[0];
            offset = temp[1];
            offset++;
            if (temp[2] == 1)
                offset++;
            to += delay;
        }
    }

    function doInsert(j3, n, txt, to, offs, pc) {
        var nl = 0;

        if (txt[n] == '#' && txt[n + 1] == 'n' &&
txt[n + 2] == '#') {
            setTimeout(function () {
                insertNewline(j3, n, offs, pc);
            }, to)
            n += 2;
            nl = 1;
        }
        else if (txt[n] == '#' && txt[n + 1] == 't'
&& txt[n + 2] == '#') {
            setTimeout(function () {
                insertTab(j3, n, offs, pc);
            }, to)
            n += 2;
        } else if (txt[n] == '#' && txt[n + 1] == 's'
&& txt[n + 2] == '#') {
            setTimeout(function () {
                insertSpace(j3, n, offs, pc);
            }, to)
            n += 2;
        }
        else {
            setTimeout(function () {
                insertOtherChar(j3, n, offs, txt[n],
pc);
            }, to)
```

```
        }

        return [n, offs, nl];
    }

    function insertNewline(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append("<br/>");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 2;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function insertTab(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' +
offs).append("    ");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }
```

```
    function insertSpace(j4, n2, offs, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append(" ");
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function insertOtherChar(j4, n2, offs, c, pc) {
        if (stop == 1 || pause == 1)
            return;
        else if (pauseCount > pc)
            return;
        var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
        window.scrollTo(0, top);
        $('#' + offs).append(c);
        node = j4;
        charAt2 = n2;
        resumeOffset = offs + 1;
        if (node == lastNode && charAt2 == lastChar)
            stopClear();
    }

    function deleted(j, charAt, timeOut, txt, offset,
pc) {
        var delay = speedChar;
        var to = timeOut;
        offset = parseInt(offset);
        for (var i = charAt; i >= 0; i--) {
```

```
            var temp = doDelete(j, i, txt, to,
offset, pc);
            i = temp[0];
            offset = temp[1];
            offset--;
            if (temp[2] == 1)
                offset--;
            to += delay;
        }
    }

    function doDelete(j3, n, txt, to, offs, pc) {
        var nl = 0;
        if (txt[n] == '#' && txt[n - 1] == 'n' &&
txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);
                document.getElementById(String(offs -
1)).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs - 1;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
            nl = 1;
        }
        else if (txt[n] == '#' && txt[n - 1] == 't'
&& txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
```

```
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
        }
        else if (txt[n] == '#' && txt[n - 1] == 's'
&& txt[n - 2] == '#') {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
            n -= 2;
        }
        else {
            setTimeout(function () {
                if (stop == 1 || pause == 1)
```

```
                    return;
                else if (pauseCount > pc)
                    return;
                var top =
document.getElementById(offs).documentOffsetTop() -
(window.innerHeight / 2);
                window.scrollTo(0, top);

document.getElementById(offs).innerHTML = "";
                node = j3;
                charAt2 = n;
                resumeOffset = offs;
                if (node == lastNode)
                    stopClear();
            }, to)
        }
        return [n, offs, nl];
    }

</script>
```

Kode Sumber 8.1 Javascript proses reka ulang

BIODATA PENULIS

Penulis bernama Muhammad Farhan Adha, dengan nama panggilan Farhan. Penulis lahir di Jakarta, 1 Juni 1993 dan merupakan anak kedua dari tiga bersaudara.

Penulis menempuh pendidikan formal di TK Islam Ruhama (1998-1999), SDS Angkasa IV (1999-2005), SMP Negeri 49 Jakarta (2005-2008), SMA Negeri 62 Jakarta (2008-2011) dan pendidikan S1 di jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya.

Di jurusan Teknik Informatika, penulis mengambil rumpun mata kuliah bidang Rekayasa Perangkat Lunak. Penulis memiliki ketertarikan di bidang Software Development Process, Basis Data, dan Competitive Programming. Penulis dalam masa kuliah aktif di berbagai organisasi antara lain Himpunan Mahasiswa Teknik Computer-Informatika (HTMC) sebagai Staff Departemen Media Informasi (2012-2013) dan Administrator Laboratorium Pemrograman. Penulis pernah menjadi koordinator asisten dosen Teknik Informatika pada mata kuliah Basis Data (2013-2014). Penulis dapat dihubungi melalui alamat surel mail.farhanadha@gmail.com.